UNIVERSITE DES SCIENCES SOCIALES DE TOULOUSE

LOGICIEL ET MATERIEL PERMETTANT DE TRAITER EN TEMPS REEL DES PROBLEMES HAUTEMENT COMBINATOIRES

THESE

présentée et soutenue le 16 avril 1976 par

Michel NIVAULT

Ingénieur Civil des Mines

pour obtenir le Doctorat de 3e Cycle

Spécialité : Mathématiques appliquées à l'Economie

MEMBRES DU JURY

Président : M. BAZERQUE

Suffragants : MM. DEVILLEBICHOT

MAHL

MOREAUX

UNIVERSITE DES SCIENCES SOCIALES DE TOULOUSE

LOGICIEL ET MATERIEL PERMETTANT DE TRAITER EN TEMPS REEL DES PROBLEMES HAUTEMENT COMBINATOIRES

THESE

présentée et soutenue le 16 avril 1976 par

Michel NIVAULT

Ingénieur Civil des Mines

pour obtenir le Doctorat de 3e Cycle

Spécialité : Mathématiques appliquées à l'Economie

MEMBRES DU JURY

Président : M. BAZERQUE

Suffragants : MM. DEVILLEBICHOT

MAHL

MOREAUX

UNIVERSITE des SCIENCES SOCIALES de TOULOUSE

PERSONNEL de l'UNIVERSITE

Année Universitaire 1975-1976

HONORARIAT

MM. MAURY, CH. L.H.	Professeur honoraire, doyen honoraire
BARRERE A.	Professeur honoraire, ex-doyen de la Faculté de Droit de PARIS
JAMES, Ch. L.H.	Professeur honoraire , Membre de l'Institut
LASSEGUE,	Professeur à l'Université de PARIS I
RAYNAUD,	Professeur à l'Université de PARIS II
VEDEL, Com. L.H.	Professeur à l'Université de PARIS II
COUZINET, O.L.H.	Professeur honoraire
ROUSSIER, Ch. L.H.	Professeur honoraire

PROFESSEURS, MAITRES de CONFERENCES AGREGES, MAITRES de CONFERENCES, CHARGES de COURS, MAITRES-ASSISTANTS.

PROFESSEURS

MM. PALLARD,	Professeur de Droit Commercial Président de l'Université
HEBRAUD, Ch.L.H.	Professeur de Droit Civil
OURLIAC, O. L.H.	Professeur d'Histoire des Institutions Membre de l'Institut
DAUVILLIER, Ch. L.H.	Professeur de Droit Romain
VIGREUX, Ch. L.H.	Professeur d'Economie Politique
CLUSEAU, Ch. L.H.	Professeur d'Economie Politique Vice-Président de l'Université
LETINIER,	Professeur d'Econométrie
BOYER,	Professeur de Droit Civil
MERLE,	Professeur de Droit Criminel
MONTANE de la ROQUE,	Professeur de Droit Constitutionnel
MARTIN de la MOUTTE,	Professeur de Droit Civil
DUPEYROUX O.	Professeur de Droit Administratif
VELLAS,	Professeur de Droit International Public
VINCENS,	Professeur de Législation Française des Finances et de Science Financière
VIDAL,	Professeur de Droit Privé
DESPAX,	Professeur de Droit Privé
SICARD,	Professeur d'Histoire du Droit
SIORAT,	Professeur de Droit Public
GILLES,	Professeur de Droit Romain
BARRERE J.	Professeur de Droit Privé
MAZERES,	Professeur de Droit Public
DEVILLEBICHOT,	Professeur de Sciences Economiques
SEMPE,	Professeur de Statistique et Méthodes d'Observation
ROUJOU de BOUBEE,	Professeur de Législation Comparée

.../...

MM. RAYBAUD,	Professeur d'Histoire des Faits Economiques et Sociaux
MOURGEON,	Professeur de Droit Public
DELVOLVE,	Professeur de Droit Public
GOUR,	Professeur de Droit Public
DAGOT,	Professeur de Droit Privé
ISAAC,	Professeur de Droit Public Vice-Président de l'Université
SALETTE,	Professeur de Sciences Economiques

PROFESSEUR ASSOCIE

M. CROS,	Professeur de Sciences Economiques

PROFESSEUR SANS CHAIRE

M. POUMAREDE,	Professeur d'Histoire des Institutions
M. SPITERI,	Professeur de Droit Privé
Melle BRUGUIERE,	Professeur d'Histoire des Institutions

MAITRES de CONFERENCES

MM. MOLINIER,	Maître de Conférences Agrégé de Droit Public
BAZERQUE,	Maître de Conférences d'Informatique
PINHAS,	Maître de Conférences de Mathématiques
ARCHER,	Maître de Conférences Agrégé de Sciences Economiques
CABANIS,	Maître de Conférences Agrégé d'Histoire des Institutions
PISTRE,	Maître de Conférences deGGestion
SERLOOTEN,	Maître de Conférences Agrégé de Droit Privé
CAPIAN,	Maître de Conférences Agrégé de Sciences Economiques
BOUYSSOU,	Maître de Conférences Agrégé de Droit Public

MAITRE de CONFERENCES ASSOCIE

M. STOKA,	Maître de Conférences de Mathématiques

CHARGES de COURS

MM. DUPEYRON,	Chargé de Cours de Droit Privé
AMADIO,	Chargé de Cours de Droit Public
CAMPAN,	Chargé de Cours de Sciences Economiques
MOREAUX,	Chargé de Cours de Sciences Economiques
Mme ALCOUFFE,	Chargé de Cours de Sciences Economiques
MM. VILLEVIEILLE,	Chargé de Cours de Droit Public
TOMASIN,	Chargé de Cours de Droit Privé
ASSARAF,	Chargé de Cours de Sciences Economiques
LE POTTIER,	Chargé de Cours de Sciences Economiques
LAVIALLE,	Chargé de Cours de Droit Public
Mme CALMETTE,	Chargé de Cours de Sciences Economiques

.../...

MAITRES-ASSISTANTS

Mme SICARD,	Maître-Assistant d'Histoire des Institutions
M. LABAUVIE,	Maître-Assistant de Sciences Economiques
M. LUDWIG,	Maître-Assistant de Droit Public
Mme CAMBOULIVES,	Maître-Assistant de Droit Privé
Mme HEUZE,	Maître-Assistant de Mathématiques
MM. ARLANDIS,	Maître-Assistant de Gestion
HERSANT,	Maître-Assistant de Sciences Economiques
COULET,	Maître-Assistant de Droit Public
MARICHY,	Maître-Assistant de Droit Public Vice-Président de l'Université
TOURNIE,	Maître-Assistant de Droit Public
LOUBET del BAYLE,	Maître-Assistant de Science Politique
AUBERT,	Maître-Assistant de Gestion
BAUX,	Maître-Assistant de Gestion
LUGAN,	Maître-Assistant de Sociologie
Mme BRUGNES,	Maître-Assistant de Sciences Economiques
MM. ARAGON,	Maître-Assistant de Mathématiques
ALBOUY,	Maître-Assistant de Science Politique
DESMOUTIER,	Maître-Assistant de Gestion
GALAN,	Maître-Assistant de Gestion
REDONNET,	Maître-Assistant d'Anglais
Mle ROUJOU de BOUBEE,	Maître-Assistant de Droit Privé
Mle GUERRIERO,	Maître-Assistant de Droit Privé
M. BIANCONI,	Maître-Assistant de Science Politique

MAITRE-ASSISTANT ASSOCIE

M. KUCERA,	Maître-Assistant de Droit Public

PERSONNEL DETACHE

MM. SIROL,	Professeur (Sciences Economiques) détaché à l'Ambassade de France à Mexico
MOLINS-YSAL,	Professeur (Sciences Economiques) détaché à Montréal
MESTRE,	Maître de Conférences Agrégé (Droit Public) détaché à Tunis

---

Mme ROULLAND,	Conseiller Administratif des Services Universitaires Secrétaire Général de l'Université

PRESIDENT de la THESE :

Suffragants :

L'Université n'entend pas approuver, ni désapprouver les opinions particulières du candidat.

Je désire exprimer ma grande reconnaissance à Monsieur Georges BAZERQUE, Professeur à l'Université des Sciences Sociales de TOULOUSE dont l'aide m'a été précieuse tout au long de la réalisation de cette thèse, et qui a bien voulu me faire l'honneur de présider ce jury.

Je tiens à assurer de ma profonde gratitude Monsieur Robert MAHL qui a dirigé mon travail et qui a accepté d'être membre de ce jury. Ce n'est que grâce à ses encouragements, ses conseils et son soutien que cette étude a pu être réalisée.

Je suis heureux d'adresser mes remerciements à Messieurs Guy DEVILLEBICHOT et Michel MOREAUX, professeurs à l'université des Sciences Sociales de TOULOUSE et membres de ce jury.

Je désire aussi remercier Monsieur Serge GUIBOUD-RIBAUD, Directeur du Département Informatique à l'Ecole des Mines de SAINT-ETIENNE, qui a bien voulu m'accueillir dans son Département pour réaliser ce travail.

Mes remerciements iront aussi à :

- M. MEYER, Directeur du Service de la Recherche Cybernétique à la SNCF et M. GENETE, ingénieur à la SNCF, qui ont accepté de me fournir de nombreux documents relatifs au combinateur optimisant et aux problèmes à tester.

- M. VUCHOT, Président-Directeur Général de la société CYBCO, qui a bien voulu confier un optimateur à l'Ecole des Mines de SAINT-ETIENNE pour en faire l'évaluation, ainsi que de nombreux documents.

- M. MONPETIT, Directeur du SESORI, qui a aidé l'Ecole des Mines à entreprendre des recherches en ordonnancement.

- MM. JULLIEN, MIREAUX, PETIT-FRESSANCOURT, ingénieurs à l'Ecole des Mines, qui, par leur collaboration étroite, m'ont permis d'avancer dans ces travaux.

- Mademoiselle M. C. MONTMARTIN, Mademoiselle B. ZOLD, MM. Roger BROSSARD et André LOUBET qui se sont chargés de la réalisation matérielle de ce document.

- Et, enfin, mon épouse qui a su m'encourager pendant la réalisation et la rédaction de ce travail.

SOMMAIRE

RESUME

Ce document examine certaines méthodes de résolution des problèmes d'ordonnancement qui font suite aux travaux du Docteur SAUVAN sur la conception de machines simulant le fonctionnement du cerveau humain lors de la résolution de problèmes hautement combinatoires.

Après avoir exposé les principes communs à toutes ces machines, il étudie l'optimateur de la société CYBCO et le combinateur optimisant de la SNCF. Le mécanisme de ces machines est exposé, à la fois sous l'angle :

1 - de la résolution de sous-problèmes à deux variables,

2 - de la recombinaison des solutions obtenues pour déterminer une solution du problème à N variables (N > 2).

Enfin dans une dernière partie, sont présentées les solutions de divers problèmes traités par ces machines et par des méthodes logicielles traditionnelles, puis les résultats obtenus sont comparés.

INTRODUCTION

Nous avons essayé au cours de cette étude, de décrire et d'évaluer les machines appelées " Calculateurs Spécialisés " conçues dans le but de résoudre les problèmes d'ordonnancement hautement combinatoires devant lesquels les ordinateurs les plus perfectionnés restent impuissants. Ces machines utilisent une méthode originale qui consiste à simuler le fonctionnement du cerveau humain lors de la résolution d'un problème complexe.

La première publication remonte à 1953 lorsque le Docteur SAUVAN présenta une communication sur le thème : Machine de simulation du processus intelligent [1]. Le Docteur SAUVAN, spécialiste de neurologie, a émis l'hypothèse que le cerveau décompose un problème complexe à N composantes en sous-problèmes à 2 ou 3 composantes, qu'il résoud simultanément les différents sous-problèmes et qu'ensuite il recompose toutes les solutions obtenues pour parvenir à la solution du problème initial.

En 1961, le Docteur SAUVAN déposa un premier brevet [2] qui conduisit en 1964 à la construction d'une machine résolvant des problèmes à deux composantes, puis d'une machine résolvant des problèmes à 3 composantes et destinée à calculer la trajectoire optimale de bombardiers volant à basse altitude, et enfin, en 1965 à la construction d'une machine pour des problèmes à 6 composantes.

En 1970, un prototype de " simulateur de mouvements de trains pour l'étude de problèmes combinatoires ferroviaires " a été livré au Service de la Recherche de la SNCF par la SNECMA. Ce prototype permettait de résoudre des problèmes à 6 composantes mais il a du être abandonné à cause de ses limitations.

Ensuite, deux équipes de recherche, l'une appartenant à la Société CYBCO, l'autre à la SNCF, s'inspirent de ces travaux en vue de mettre au point une machine qui puisse apporter une solution en temps réel aux problèmes d'ordonnancement dans lesquels intervient un grand nombre de composantes. C'est à la suite des recherches de ces deux équipes que se placent nos travaux.

La société CYBCO, constituée en 1972 dans le but d'exploiter les brevets SAUVAN, réalise en 1973 une machine résolvant des problèmes à deux composantes et appelée " optimateur ", de conception beaucoup plus légère que toutes les machines précédemment construites. Cet optimateur fut installé pendant quelques mois dans le Centre de Calcul de l'Ecole des Mines de Saint-Etienne, ce qui permit d'une part de le tester et d'autre part de lui adjoindre une partie logicielle grâce au couplage à un ordinateur général (non spécialisé) en vue de traiter des problèmes ayant de nombreuses composantes. Après que la société CYBCO eut repris l'optimateur, nous avons simulé son fonctionnement sur ordinateur afin de poursuivre notre étude.

De son côté, Monsieur GENETE, ingénieur au Service de la Recherche de la SNCF présenta en 1971 une thèse [3] sur la réalisation d'une machine servant à régulariser le trafic ferroviaire. Cette machine fut ensuite simulée sur ordinateur à la SNCF et testée sur des problèmes d'affectation et d'ordonnancement. Dans le cadre de notre étude, une simulation du principe général du fonctionnement de ce calculateur a été réalisée et testée sur ordinateur à l'Ecole des Mines.

Le chapitre 1 de cette thèse a pour but de définir les problèmes d'ordonnancement en temps réel, de préciser les notations, les classes de problèmes et les critères d'optimisation couramment rencontrés.

Dans un second chapitre, nous étudions les principes généraux des calculateurs spécialisés, nous montrons comment l'idée originale du Docteur SAUVAN a pu permettre de résoudre des problèmes d'ordonnancement en se ramenant à la recherche d'une trajectoire optimale dans un espace à N dimensions (N>2).

Nous trouvons au chapitre 3 la description de l'algorithme de l'optimateur CYBCO qui permet de résoudre des problèmes à deux composantes. Ensuite nous exposons les difficultés rencontrées pour faire fonctionner l'optimateur dans certains cas de figure. Enfin, nous présentons le logiciel qui a du être couplé à la machine CYBCO pour résoudre les problèmes à N composantes.

Le quatrième chapitre est un exposé de l'algorithme principal du combinateur SNCF que nous avons simulé sur ordinateur.

Enfin, dans un dernier chapitre nous présentons divers résultats obtenus en résolvant certains problèmes d'ordonnancement selon les méthodes de ces calculateurs spécialisés et selon des méthodes purement logicielles.

1 - LES PROBLEMES D'ORDONNANCEMENT " TEMPS REEL ".

1 - 1. POSITION DU PROBLEME [4]

Nous appellerons processus une succession de tâches à exécuter. Il y a, dans un problème d'ordonnancement, d'une part des contraintes sur l'ordre d'exécution des tâches de chaque processus, et d'autre part des contraintes d'exclusion mutuelle entre les exécutions de diverses tâches de certains processus. Compte tenu de toutes ces contraintes ainsi que des contraintes de temps puisque chaque tâche est de durée définie à l'avance, le problème consiste à déterminer l'ordre et la date des débuts d'exécution de chaque tâche. Cet ordre doit être établi en fonction d'un critère que l'on cherche à minimiser. Il peut être défini comme une permutation $(\alpha_1, \alpha_2, \ldots\ldots, \alpha_n)$ des entiers $(1, 2, \ldots\ldots, n)$ où n est le nombre total des tâches pour l'ensemble des processus qui entrent dans le problème.

Nous utiliserons comme synonymes les termes variables et processus, nous montrerons sur des exemples pratiques que des trains peuvent être des processus, de même que des pièces dont il s'agit d'ordonnancer la fabrication dans un atelier.

Comme nous allons le voir, les problèmes d'ordonnancement se retrouvent dans de nombreux secteurs du monde industriel d'aujourd'hui. De leur résolution satisfaisante ou non dépendent souvent la bonne marche de l'entreprise et l'accroissement de la productivité. Malheureusement, ils sont très difficiles à résoudre pour les raisons suivantes :

- ils sont souvent très combinatoires (de nombreux processus sont en jeu),
- les contraintes ne sont pas toujours connues avec précision,
- le critère en fonction duquel est déterminé l'ordonnancement, est souvent choisi de façon subjective dans un ensemble de critères qui peuvent être contradictoires.

1-1-1- Ordonnancement d'atelier :

Il s'agit d'organiser au mieux le fonctionnement d'un atelier. Les processus sont ici des pièces, l'atelier est constitué de plusieurs machines, chaque machine doit exécuter un certain nombre de tâches, chaque tâche étant relative à une pièce donnée. Nous savons en outre que :

- chaque pièce est traitée sur les différentes machines selon un ordre déterminé par sa gamme de fabrication,
- la durée d'exécution de chaque tâche, constituée par le passage d'une pièce sur une machine, est connue avec précision à priori.

Les contraintes de ce problème sont le plus souvent :

- une machine ne peut traiter qu'une pièce à la fois,
- dès qu'une tâche est commencée sur une machine, elle doit être terminée avant qu'une nouvelle tâche ne soit entreprise sur la même machine,
- dans certains cas, il est nécessaire d'introduire un délai de réemploi d'une machine entre deux tâches (temps nécessaire au changement d'outil par exemple).

Le but de l'ordonnancement est de déterminer, pour chaque machine, l'ordre dans lequel elle doit usiner les différentes pièces. Cet ordre pourra être différent selon le critère en fonction duquel on cherchera la solution optimale, faut-il :

- chercher à minimiser le temps global d'occupation de l'atelier,
- chercher à minimiser les retards de fabrication à la sortie de l'atelier,
- etc .....

1-1-2- Ordonnancement des travaux dans un ordinateur multiprogrammé :

Aujourd'hui, la plupart des gros ordinateurs partagent leur activité entre différents travaux selon diverses méthodes :

- dans le cas d'un système à multiprogrammation classique, les processus sont des travaux, plusieurs travaux qui résident en mémoire centrale sont en concurrence pour obtenir les services de l'unité centrale ou des périphériques. Un problème d'ordonnancement se pose pour déterminer l'ordre d'exécution de ces travaux. Deux critères différents peuvent également être utilisés pour définir cet ordonnancement: optimisation de l'utilisation globale des ressources ou avancement pondéré de l'ensemble des travaux,

- dans le cas d'un système "temps réel", chaque processus est réactivé régulièrement par une horloge. On dispose alors d'un certain délai pour attribuer l'unité centrale au processus réactivé. L'algorithme le plus couramment utilisé est le "shortest deadline first", (priorité au plus court délai), qui alloue l'unité centrale au processus dont la fin du délai est la plus rapprochée. L'intérêt de cet algorithme a été montré par J. Labetoulle [5],

- dans le cas d'un système en temps partagé, l'ordonnancement des processus tient compte de divers critères pondérés, comme la longueur probable de la tâche qui démarre, son temps d'activité passée, etc ...

- dans un réseau d'ordinateurs à commutation de paquets, un paquet est un processus et son envoi d'un commutateur à un autre est une tâche ; chaque commutateur doit décider très rapidement des paquets à envoyer et des lignes à utiliser. Les critères d'optimisation choisis sont les priorités des paquets et le décongestionnement du réseau.

Vu la rapidité d'un ordinateur, toutes ces décisions doivent être prises dans des temps de l'ordre du centième de seconde et quelquefois moins, ce qui montre la nécessité de déterminer très rapidement l'ordonnancement des tâches.

1-1-3- *Problèmes de trafic* :

Dans le monde moderne, les problèmes de communications prennent chaque jour davantage d'importance, les individus ou les groupes d'individus ont de plus en plus besoin d'être souvent en contact, de se déplacer, d'échanger des idées. Or, pour que ces échanges aient lieu, il faut un support, une infrastructure. Pour répondre aux besoins sans cesse grandissants, on est tenté de multiplier les réseaux de communications, ce qui est fort coûteux et conduit à une sous-utilisation des réseaux. On peut aussi chercher à organiser le mieux possible l'écoulement du trafic sur les réseaux existants, ce qui permet d'accroître les échanges sans augmenter l'infrastructure.

Pour appliquer cette dernière solution, il sera indispensable de pratiquer des ordonnancements. Des améliorations pourront ainsi être apportées dans des domaines aussi divers que la régulation du trafic routier, ferroviaire, aérien, maritime ou téléphonique.

1 - 2. NOTATION DANS LE CAS DU PROBLEME D'ORDONNANCEMENT D'ATELIER [6]

Remarquons que les trois types de problèmes que nous venons d'envisager peuvent se ramener assez facilement à l'ordonnancement d'atelier. En effet, l'usinage de pièces sur une machine obéit à des contraintes et à des critères d'optimisation, de la même façon que des tâches dans l'unité centrale d'un ordinateur, ou que des mobiles sur des portions de routes, de voies ou de couloirs aériens. Dans tous les cas, on dispose de ressources limitées sur lesquelles (ou à l'aide desquelles), il faut faire évoluer des processus.

Pour clarifier l'exposé, nous ne parlerons désormais dans ce chapitre que d'ordonnancement d'atelier. L'intérêt de cette convention est de pouvoir ainsi aborder les problèmes d'ordonnancement de façon très concrète tout en définissant une notation précise.

Nous pouvons dès lors poser le problème de la manière suivante :

" Etant données "n" pièces (ou processus) qui doivent être traitées sur "m" machines dans un ordre donné et sous certaines contraintes, quel est, relativement à un critère donné, l'ordre optimal dans lequel chaque machine traite l'ensemble des pièces ? "

Au premier abord, un problème est donc défini par un ensemble de données :

- n pièces P_1, P_2, ..., P_n,
- m machines M_1, M_2, ..., M_m,
- une pièce P_k et une machine M_ℓ déterminent de façon unique une tâche désignée par (P_k, M_ℓ) ou plus simplement (k, ℓ). La durée d'exécution de cette tâche est notée $d_{k\ell}$. Chaque processus (fabrication de pièces) se compose d'un certain nombre de tâches dont I'ordre est déterminé à l'avance. Cet ordre constitue la gamme de la pièce.
- $G(i,j) = \ell$ signifie que la $j^{\text{ème}}$ machine de la gamme de P_i est M_ℓ.

Conway, Maxwell et Miller [4] caractérisent chaque type de problème par un ensemble de valeurs selon la notation suivante :

$$A|B|C|D_1, D_2, ...|E$$

où :

- A est le nombre de pièces,
- B est le nombre de machines,
- C est le type de gammes de fabrication. On en distingue essentiellement trois :

. (G1) : toutes les gammes sont identiques (flow-shop). Cela ne veut pas dire que les machines traitent nécessairement les pièces dans le même ordre.

. (G2) : toutes les gammes sont identiques et en outre les machines traitent les pièces dans le même ordre.

. (G3) : les gammes sont différentes (job-shop).

- D_1, D_2, ..., D_n ensemble d'hypothèses restrictives ou contraintes qui caractérisent le problème d'ordonnancement.
- E : critère à optimiser.

Envisageons maintenant de façon plus détaillée les contraintes et les critères servant à définir un problème d'ordonnancement.

1-2-1- Hypothèses restrictives ou contraintes :

Les hypothèses restrictives ou contraintes les plus fréquemment rencontrées sont les suivantes :

(P1) l'ensemble ($\mathcal{P}$) des pièces est connu et fixé.
(M1) l'ensemble ($\mathcal{M}$) des machines est connu et fixé.
(P2) toutes les pièces sont disponibles au même instant t = 0.
(M2) toutes les machines sont disponibles au même instant t = 0.
(P3) toutes les pièces restent disponibles pendant un temps infini (pas de dates exigées).
(M3) toutes les machines restent disponibles pendant un temps infini (pas de grèves ni de pannes).
(P4) chaque pièce est dans l'un des trois états suivants : attendant sa prochaine machine, en traitement sur une machine, ou ayant quitté la dernière machine de sa gamme.
(M4) chaque machine est dans l'un des trois états suivants : attendant la prochaine pièce, travaillant sur une pièce ou ayant terminé sa dernière pièce.
(P5) toutes les pièces sont différentes.
(M5) toutes les machines sont différentes.
(P6) toutes les pièces sont également importantes.
(M6) toutes les machines sont également importantes.
(P7) chaque pièce passe sur toutes les machines de sa gamme.
(M7) chaque machine traite toutes les pièces qui lui sont assignées.
(P8) chaque pièce occupe une seule machine à la fois.
(M8) chaque machine traite une pièce à la fois.
(PM1) toutes les durées d'exécution sont connues et fixées (c'est-à-dire indépendantes de l'ordre dans lequel les pièces passent sur les machines ; autrement dit pas de durées de réemploi).
(PM2) toute tâche commencée doit aller jusqu'à son complet achèvement sans être interrompue, (non préemptibilité).
(PM3) les gammes de fabrication sont connues et fixées.
(PM4) l'ordre de passage des pièces sur chaque machine est connu et fixé.

Certaines de ces hypothèses sont beaucoup plus importantes que d'autres.

(P2) et (M2) peuvent être rejetées sans que cela ne modifie de manière significative le problème posé et donc les méthodes susceptibles de le résoudre. Par contre (P1) et (M1) sont fondamentales car elles font la distinction entre les problèmes statiques (déterministes) et les problèmes dynamiques (stochastiques).

Trois exemples de caractérisation de problèmes :

Exemple 1 :

n pièces doivent être fabriquées par une machine. On fait intervenir des durées de réemploi. Un tel problème, où l'hypothèse (PM1) n'est pas valide, sera décrit par :

$$n\,|\,1\,|\,(PM1)\,|\,E$$

où E est un critère d'optimalité. Ici, les hypothèses relatives aux gammes n'interviennent pas puisque la gamme est unique.

C'est typiquement le problème du "voyageur de commerce".

Exemple 2 :

Même problème mais avec m machine et des gammes de type G2. On obtient :

$$n\,|\,m\,|\,G2\,|\,(PM1)\,|\,E$$

Cette formulation est celle du problème que nous avons désigné plus loin sous le nom de "trainsau départ".

Exemple 3 :

n tâches, m machines. Toutes les gammes sont différentes. On rejette l'hypothèse (PM2) de non préemptibilité.Le critère est une optimisation de moyenne pondérée.

$$n\,|\,m\,|\,G3\,|\,(P6),(PM2)\,|\,E$$

1-2-2- *Critères d'optimisation* :

On peut les classer en cinq groupes principaux :

- critères basés sur les dates de terminaison et les temps de passage.
- critères basés sur les dates exigées.
- critères basés sur les notions d'en-cours et de coefficient d'utilisation.
- critères basés sur les durées de réemploi.
- critères multiples obtenus en combinant tout ou partie des précédents.

1-2-2-1- *Critères basés sur les dates de terminaison et les temps de passage* :

C_k = date à laquelle la pièce P_k est disponible (date au plus tôt à partir de laquelle elle peut être mise en fabrication). Si l'hypothèse (P2) est admise, $C_K=0, \forall K$.

$a_{k\ell}$ = temps d'attente de la pièce P_k devant la machine M_ℓ.

$A_k = \sum_{\ell=1}^{m} a_{k\ell}$ temps d'attente total de P_k.

$d_k = \sum_{\ell=1}^{m} d_{k\ell}$ durée d'exécution totale de P_k.

T_k = date de terminaison de P_k.

F_k = temps total que P_k passe dans l'atelier.

Il existe des relations élémentaires entre ces grandeurs :

$$T_k = C_k + A_k + d_k \qquad (1)$$

$$F_k = A_k + d_k \qquad (2)$$

Nous pouvons maintenant définir les critères les plus fréquemment utilisés :

(C1) Minimiser le maximum des dates de terminaison ou encore le temps total d'activité de l'atelier : $T_{max} = \max_k \{T_k\}$.

(C2) Minimiser le maximum des durées de passage : $F_{max} = \max_k \{F_k\}$.

(C3) Minimiser le maximum des temps d'attente : $A_{max} = \max_k \{A_k\}$.

Trois critères portant sur les moyennes pondérées, le coefficient α_k affecté à la pièce K mesure son importance, si $\alpha_k = 1, \forall k$, l'hypothèse (P6) est respectée, sinon elle est rejetée :

(C4) Minimiser la moyenne pondérée des dates de terminaison : $\sum_k \alpha_k T_k$.

(C5) Minimiser la moyenne pondérée des durées de passage : $\sum_k \alpha_k F_k$.

(C6) Minimiser la moyenne pondérée des temps d'attente : $\sum_k \alpha_k A_k$.

Les égalités (1) et (2) montrent que les critères (C4), (C5),(C6) sont équivalents.

Les critères (C1) et (C2) sont équivalents si $C_k = 0, \forall k$, ce que nous supposerons désormais sauf en cas de convention explicite contraire.

1-2-2-2- *Critères basés sur les dates exigées* :

Ces critères imposent le rejet de l'hypothèse (P3).

f_K = date exigée pour la fin d'exécution de P_K.

$Q_K = T_K - f_K$, retard algébrique de P_K.

$R_K = MAX(0, Q_K)$, retard réel de P_K.

$S_K = MAX(0, -Q_K)$, avance réelle de P_K.

Les critères utilisés sont alors les suivants :

(C7) minimiser $Q_{max} = \max_k \{Q_k\}$

(C8) minimiser $R_{max} = \max_k \{R_k\}$

(C9) maximiser $S_{max} = \max_k \{S_k\}$

(C10) minimiser $\sum_k \alpha_k Q_k$

(C11) minimiser $\sum_k \alpha_k R_k$

(C12) maximiser $\sum_k \alpha_k S_k$.

On a par définition :

$$\sum_k \alpha_k Q_k = \sum_k \alpha_k T_k - \sum_k \alpha_k f_k .$$

Ce qui montre que (C10) est équivalent à (C4), (C5), (C6).

1-2-2-3- *Critères basés sur les notions d'en-cours et de coefficient d'utilisation* :

Il est possible de juger la qualité d'un ordonnancement par l'examen attentif de ce qui se passe tout au long du processus de production. On introduit :

Np(t) : nombre de pièces terminées à l'instant t,
Na(t) : nombre de pièces en attente devant une machine à l'instant t,
Nf(t) : nombre de pièces en cours de fabrication à l'instant t,
Wp(t) : travail exécuté, c'est-à-dire somme des temps d'exécution des tâches terminées à l'instant t,
Wa(t) : travail restant à faire, somme de temps d'exécution des tâches restant à effectuer à l'instant t,
Wf(t) : travail en cours, somme des temps des tâches en cours d'exécution à l'instant t.

Par définition :

$$Np(t) + Na(t) + Nf(t) = n$$

$$Wp(t) + Wa(t) + Wf(t) = \sum_{k,\ell} d_{k\ell}$$

Si on considère maintenant les moyennes de ces grandeurs prises sur l'intervalle (0, Fmax), on voit que :

- $\overline{Na} + \overline{Nf}$ est le niveau moyen des en-cours,
- $\overline{Np}$ est le niveau moyen des produits finis.

Les critères proposés sont alors les suivants :

(C13) maximiser $\overline{Np}$ niveau moyen des produits finis.
(C14) minimiser $\overline{Na} + \overline{Nf}$ niveau moyen des en-cours.
(C15) minimiser $\overline{Na}$ nombre moyen de pièces en attente.
(C16) maximiser $\overline{Nf}$ nombre moyen de pièces en cours de fabrication.
(C17) maximiser $\overline{Wf}$ moyenne du temps d'exécution des tâches en cours.
(C18) minimiser la somme des temps morts : $m\, F_{max} - \sum_{k,\ell} d_{k\ell}$.

(C19) maximiser le coefficient d'utilisation $\overline{U} = \dfrac{\sum_{k,\ell} d_{k\ell}}{m\, F\, max}$.

1-2-2-4- *Critères basés sur les durées de réemploi* :

On considère un problème à n pièces et une machine où, à la durée d'exécution de la pièce k, il faut ajouter le temps d'adaptation ou de réemploi de la machine lorsqu'elle passe d'une pièce j à la pièce k. Autrement dit, la durée d'exécution d'une tâche dépend de la séquence. Le seul critère utilisé est la minimisation des durées de réemploi.

1-2-2-5- Critères multiples :

Ces critères sont obtenus en combinant plusieurs critères (Ci) définis précédemment.

1-2-2-6- Récapitulation :

Comme nous l'avons montré pour quelques-uns, certains de ces critères sont équivalents entre eux. Il est donc possible de les regrouper en classes que nous caractériserons par le critère le plus simple à formuler :

(C1), (C2), (C16), (C17), (C18), (C19) : Minimisation du maximum des durées de passage Fmax,

(C3) Minimisation du maximum des temps d'attente Amax ,

(C4), (C5), (C6), (C10) Minimisation de la moyenne pondérée des durées de passage $\Sigma\alpha_k F_k$,

(C7) Minimisation du maximum du retard algébrique Qmax,

(C8) Minimisation du maximum du retard Rmax,

(C9) Maximisation du maximum de l'avance Smax,

(C11) Minimisation de la moyenne pondérée du retard $\Sigma\alpha_k R_k$,

(C12) Maximisation de la moyenne pondérée de l'avance $\Sigma\alpha_k S_k$,

(C13), (C14) Maximisation du nombre moyen de pièces finies $\overline{Np}$,

(C15) Minimisation du nombre moyen de pièces en attente $\overline{Na}$.

1 - 3. LA CONTRAINTE TEMPS REEL :

Le temps disponible, dans la pratique, pour résoudre un problème d'ordonnancement est très variable et dépend du problème traité.

S'il s'agit d'organiser le programme de fabrication mensuel d'un atelier ou d'établir les horaires des trains pour les mois à venir, le problème peut être abordé bien avant que la solution ne soit utilisée sur le terrain. Le temps disponible pour la recherche de cette solution n'est limité que par le coût de fonctionnement du programme. Il est alors possible de trouver la solution optimale à l'aide d'un algorithme d'optimisation classique comme la recherche en arbre si le nombre de solutions n'est pas trop élevé.

Par contre dans certains cas, une solution au problème posé doit être trouvée presque aussitôt après avoir lancé le programme ; il s'agit surtout des cas où interviennent des perturbations : indisponibilité subite d'une machine dans un atelier, lancement en fabrication de commandes urgentes non prévues dans la planification, prise en compte de trains supplémentaires à partir d'un planning déjà établi, circulation momentanée des trains dans les deux sens sur une même voie, embouteillage à certains carrefours, etc... Dans tous ces problèmes, il importe de trouver une "solution de rechange" (un nouvel ordonnancement) suffisamment rapidement afin de ne pas introduire des retards supplémentaires sur le terrain. C'est ce que nous appellerons l'obligation de travailler en temps réel.

En général, les problèmes d'ordonnancement sont très combinatoires. Pour un problème de type n/m/G3, le nombre de solutions compatibles avec les contraintes est fourni par la formule $(n!)^m$. Comme nous le remarquons sur le tableau suivant, ce nombre croît très rapidement avec n et m.

n	m	nombre de solutions
5	3	$1{,}7.\ 10^{6}$
10	3	$4{,}8.\ 10^{19}$
10	10	$4\ .\ 10^{63}$
100	20	10^{3160}

Le temps de résolution d'un problème par des méthodes explorant toutes les solutions est, approximativement, proportionnel au nombre de solutions possibles. Il est bien évident que, dans la plupart des cas, ce temps n'est pas compatible avec la contrainte temps réel, il est donc indispensable de rechercher d'autres méthodes.

1 - 4. APPROCHES CABLEES :

Nous appellerons heuristique tout algorithme qui permet de trouver assez rapidement une solution du problème qui soit compatible avec les contraintes et que l'on peut espérer assez proche de l'optimum sans pouvoir néanmoins le démontrer.

Les méthodes à envisager pour résoudre un problème hautement combinatoire ne peuvent être le plus souvent qu'heuristiques ; on ne peut alors qu'espérer obtenir en un temps raisonnable (compatible avec la contrainte temps réel) une "bonne solution", généralement assez proche de l'optimale.

Pour accélérer encore le temps de réponse à un problème posé, des chercheurs ont pensé câbler certaines de ces heuristiques et construire de cette façon des "calculateurs spécialisés". C'est ce domaine de l'ordonnancement que nous tentons de cerner dans cette étude.

2 - PRINCIPE DES CALCULATEURS SPECIALISES POUR LES PROBLEMES COMBINATOIRES.

2 - 1. HISTORIQUE [3] :

Un certain nombre de chercheurs ont tenté d'aborder les problèmes combinatoires en simulant sur des machines les comportements naturels : instinctif ou intelligent. Nous pouvons citer les réalisations suivantes :

- Les tortues de GREY WALTER (1948).

Ce sont des appareils munis de cellules photo-électriques commandant un moteur, ce qui leur permet de se diriger vers les sources lumineuses. De plus, ces tortues sont capables de contourner les obstacles qu'on leur présente grâce à l'établissement d'un circuit électrique alimentant le moteur lors du contact avec l'obstacle.

- Le perceptron de ROSENBLATT.

C'est un outil qui pouvait être utilisé en reconnaissance des formes car il a la propriété d'éliminer les caractères secondaires du domaine étudié pour n'en conserver que les composantes principales.

- Les matrices de STEINBUCH.

Il s'agissait de représenter sous forme matricielle et de façon extrêmement succinte les fonctions de généralisation, d'association et d'imagination de la pensée humaine.

Les premières études réalisées en France sur les machines capables de résoudre les problèmes combinatoires ont été menées autour des années 1950 par des neurophysiologues. Ces chercheurs ont imaginé, sous la direction du Docteur SAUVAN, une machine simulant le fonctionnement du cerveau humain. L'hypothèse de base est que le cerveau ne peut résoudre globalement les problèmes complexes où la solution dépend des valeurs prises par un grand nombre de variables. Ceci est mis en évidence en considérant le nombre relativement limité des neurones du cerveau (10^{20}) et le nombre nécessaire pour mémoriser l'ensemble des permutations pour un problème moyennement complexe mettant en jeu 300 situations (10^{500}, soit 300 !). On suppose que le cerveau décompose un tel problème en différents sous-problèmes très simples traités en parallèle et que, dans une dernière phase, il combine les différentes solutions partielles obtenues pour en déduire sa décision finale. C'est ce procédé qui fut à l'origine des recherches qui ont conduit à la conception des deux calculateurs spécialisés que nous étudierons aux chapitres 3 et 4.

2 - 2. *RESOLUTION DES PROBLEMES D'ORDONNANCEMENT PAR ANALYSE SPATIALE : PRINCIPE.*

L'idée fondamentale de cette étude est que la recherche de l'ordonnancement optimal de N processus se ramène à la détermination du chemin le plus court entre deux points d'un espace à N dimensions. Pour plus de clarté, nous allons d'abord exposer un problème très simple à deux processus, sa résolution peut donc être effectuée dans un plan.

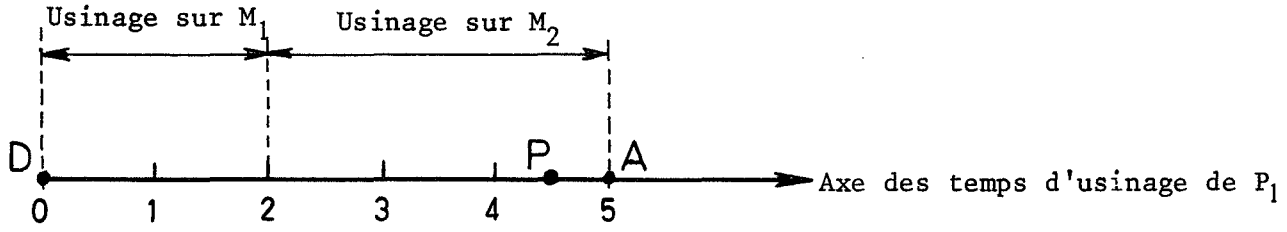
2-2-1- *Etude d'un problème à 2 processus* :

Considérons le problème d'ordonnancement suivant [7] :

- une pièce P_1 doit être usinée successivement :
  - . pendant 2 heures sur la machine M_1,
  - . pendant 3 heures sur la machine M_2,

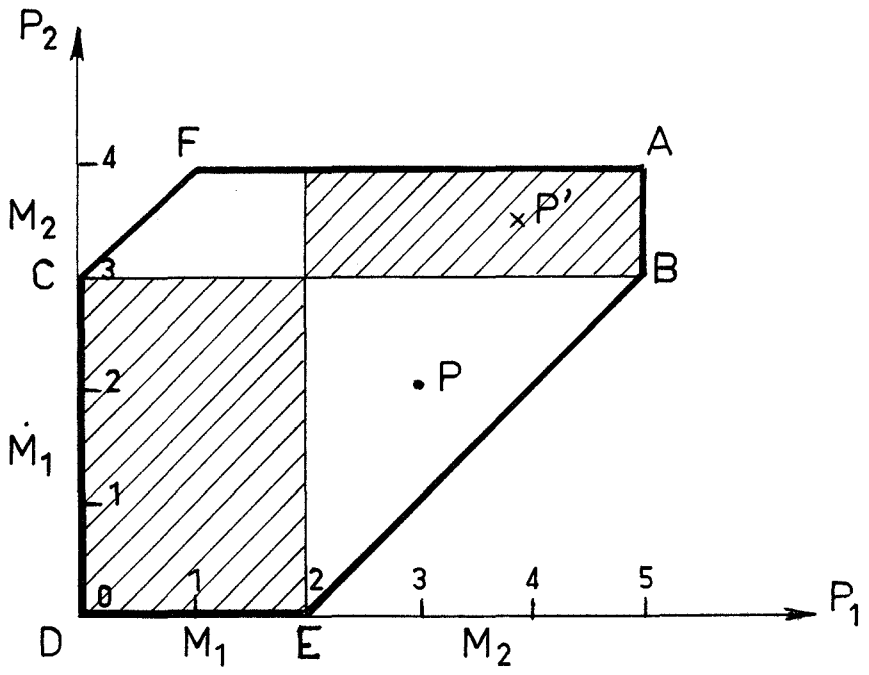
- une pièce P_2 doit être usinée successivement :
  - . pendant 3 heures sur la machine M_1,
  - . pendant 1 heure sur la machine M_2.

Selon la notation définie en 1-2, il s'agit d'un problème 2|2|(G1)|Fmax. Graphiquement, il est possible de représenter l'état d'avancement de l'usinage de la pièce P_1 par un point P sur l'axe des temps d'usinage de P_1.



La position de P indique que P_1 a fini d'être usinée sur M_1 et que son usinage sur M_2 se terminera après encore une demi-heure de travail. Lorsque P sera en A, la pièce P_1 sera terminée. Insistons dès maintenant sur le fait que l'axe ci-dessus représente les temps réels d'usinage de P_1, c'est-à-dire qu'il se peut très bien qu'au cours de la fabrication de P_1 on ait, pour une raison quelconque, décidé de laisser la pièce en attente pendant un certain temps et ce temps n'est pas représenté sur l'axe. Ceci va être éclairé par la représentation simultanée de l'usinage des deux pièces P_1 et P_2 qui est faite maintenant.

Considérons un repère orthonormé, l'axe des abscisses représentant l'usinage de P_1 et l'axe des ordonnées, celui de P_2.



Soit un point P quelconque du plan, son abscisse représente le temps d'usinage déjà réalisé de P_1 et son ordonnée celui de P_2. Cependant, le respect de la contrainte de non préemptibilité (PM2) interdit au point P de se trouver dans certaines régions. En effet, si P était en P', cela signifierait que les deux pièces sont simultanément en cours de traitement sur la même machine M_2, ce que nous avons supposé irréalisable. Il existe donc dans le plan des régions interdites (hachurées sur la figure), de forme rectangulaire, que nous appellerons "obstacles" ou "contraintes". L'unité de temps, ici l'heure, sera appelée "pas". La réalisation de cet ordonnancement, c'est-à-dire l'usinage des deux pièces P_1 et P_2, peut être figurée dans le repère par un trajet du point P reliant l'état initial D à l'état final A, en contournant les obstacles.

Examinons deux ordonnancements (ou trajets) possibles :

- Ordonnancement 1 :

. P_1 passe sur M_1 pendant 2 heures : trajet D → E.
. P_1 passe sur M_2 pendant 3 heures et en même temps
  P_2 passe sur M_1 pendant 3 heures : trajet E → B.
. P_2 passe sur M_2 pendant 1 heure : trajet B → A.

Le temps (ou coût) de l'ordonnancement 1 ainsi construit est le nombre de pas du trajet figuratif : ici 6.

- Ordonnancement 2 :

. P_2 passe sur M_1 pendant 3 heures : trajet D → C.
. P_2 passe sur M_2 pendant 1 heure et en même temps
  P_1 passe sur M_1 pendant 1 heure : trajet C → F.
. P_1 passe sur M_1 pendant 1 heure puis
  sur M_2 pendant 3 heures : trajet F → A.

Nombre de pas du trajet : 8.

L'ordonnancement 1 est donc le meilleur. Sur cet exemple simple, on constate que la recherche de l'ordonnancement optimal est équivalente à celle du trajet le plus court dans un plan.

2-2-2- *Etude d'un problème à N processus* :

D'un point de vue théorique, on conçoit dès lors qu'il est possible de résoudre un problème à N processus dans un espace N dimensionnel ; chaque axe représente le temps d'activité passée de l'un des N processus ; les obstacles (ou contraintes entre deux processus) sont figurés par des hypervolumes de base rectangulaire dans le plan relatif aux deux processus concernés et ont N-2 arêtes de longueur infinie et parallèles à chacun des autres axes. La détermination du chemin le plus court dans un tel espace fournit la solution optimale au problème posé.

Le câblage de cette méthode sur un calculateur spécialisé implique la construction physique de l'espace de recherche. Pour un problème moyennement complexe de 50 processus à ordonnancer et tel que l'avancement de chacun d'eux soit discrétisé en 1000 pas sur son axe, l'espace d'étude comporte $(10^3)^{50} = 10^{150}$ points. Cette approche a du être abandonnée car elle conduirait à la construction de machines gigantesques de coût prohibitif. Mais en utilisant l'analogie avec les hypothèses de fonctionnement du cerveau humain, une heuristique dérivée de la méthode précédente a été mise au point pour résoudre les problèmes d'ordonnancement à N processus ($N > 2$).

On se limite à la construction physique d'un plan, ce qui permet de trouver l'ordonnancement optimal de problèmes à deux processus. A partir d'un problème à N processus, il est possible de constituer $\frac{N(N-1)}{2}$ couples différents de processus. Pour chaque couple, on cherche les ordonnancements optimaux selon la représentation plane décrite en 2-2-1. Bien qu'il n'y ait aucun moyen théorique parfait pour déterminer un ordonnancement optimal de N processus à partir des ordonnancements optimaux de 2 processus, il est possible d'utiliser différents algorithmes heuristiques dits "algorithmes de recombinaison" qui fournissent de bonnes solutions du problème à N processus grâce à l'examen attentif des $\frac{N(N-1)}{2}$ représentations planes des couples de processus.

Considérons un exemple avec N = 3. Soient les pièces P_1, P_2, P_3 ; P_1 et P_2 sont les pièces définies en 2-2-1 et P_3 est une pièce dont l'exécution nécessite 7 unités de temps sur une machine M_3. Nous avons représenté cet exemple avec ses contraintes dans l'espace de dimension 3. D est le point représentant le début de l'ordonnancement, A la fin. A_{13}, A_{23} et A_{12} sont les projections de A sur les 3 plans.

La méthode utilisée pour résoudre ce problème se décompose en 2 étapes :

Etape 1 : analyse plane ou bidimensionnelle.

On considère tous les couples de processus que l'on peut former, (avec N processus, il est possible de former $\frac{N(N-1)}{2}$ couples), et pour chacun de ces couples on détermine le coût du meilleur ordonnancement, pour un premier pas imposé, en recherchant le trajet le plus court dans le plan de chaque couple.

Dans le plan P_1P_2, il existe 3 façons possibles pour construire le premier pas de la trajectoire :

1 - faire usiner P_1 seul (point X_1),
2 - faire usiner P_2 seul (point X_2),
3 - ne rien faire (point D) ; la quatrième façon : usinage simultané de P_1 et P_2 (point X_3) est impossible (pénétration dans une contrainte).

On détermine alors pour chacune de ces 3 façons les coûts C_1, C_2 et C_3 des meilleurs ordonnancements.

Dans le plan P_1P_3, il existe 4 façons possibles pour construire le premier pas de la trajectoire :

1 - faire usiner P_1 seul (point X_1),
2 - faire usiner P_3 seul (point X_4),
3 - ne rien faire (point D),
4 - faire usiner simultanément P_1 et P_3 (point X_5).

On obtient ainsi les coûts C'_1, C'_2, C'_3 et C'_4 des meilleurs ordonnancements pour chacune des 4 façons.

De même, dans le plan P_2P_3, on obtient les coûts C''_1, C''_2, C''_3 et C''_4.

Etape 2 : recombinaison.

En traitant les valeurs des différents coûts obtenus à l'étape 1 selon des méthodes heuristiques définies aux paragraphes 3-4 et 4-2, on définit un premier pas de la trajectoire spatiale. Soit DX_5 ce premier pas, il correspond à l'usinage simultané de P_1 et P_3 alors que P_2 est en attente.

X_5 constitue alors le nouveau point de départ de l'ordonnancement.

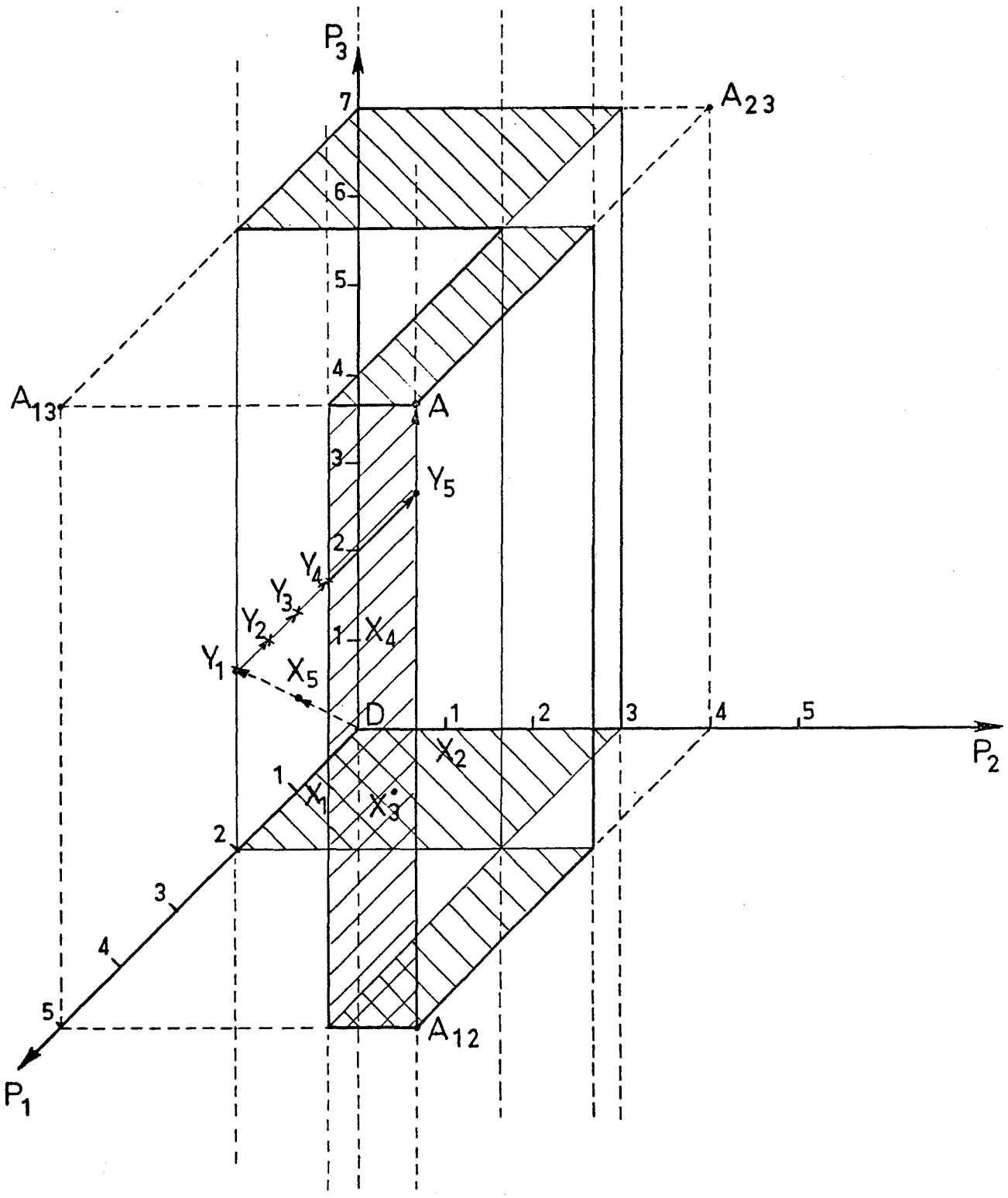
On retourne ensuite à l'étape 1 jusqu'à ce que la trajectoire soit complètement déterminée.

Dans l'exemple traité, les points de la trajectoire successivement déterminés seraient :

. D (0,0,0)
. X_5 (1,0,1)
. Y_1 (2,0,2)
. Y_2 (3,1,3)
. Y_3 (4,2,4)
. Y_4 (5,3,5)
. Y_5 (5,4,6)
. A (5,4,7)

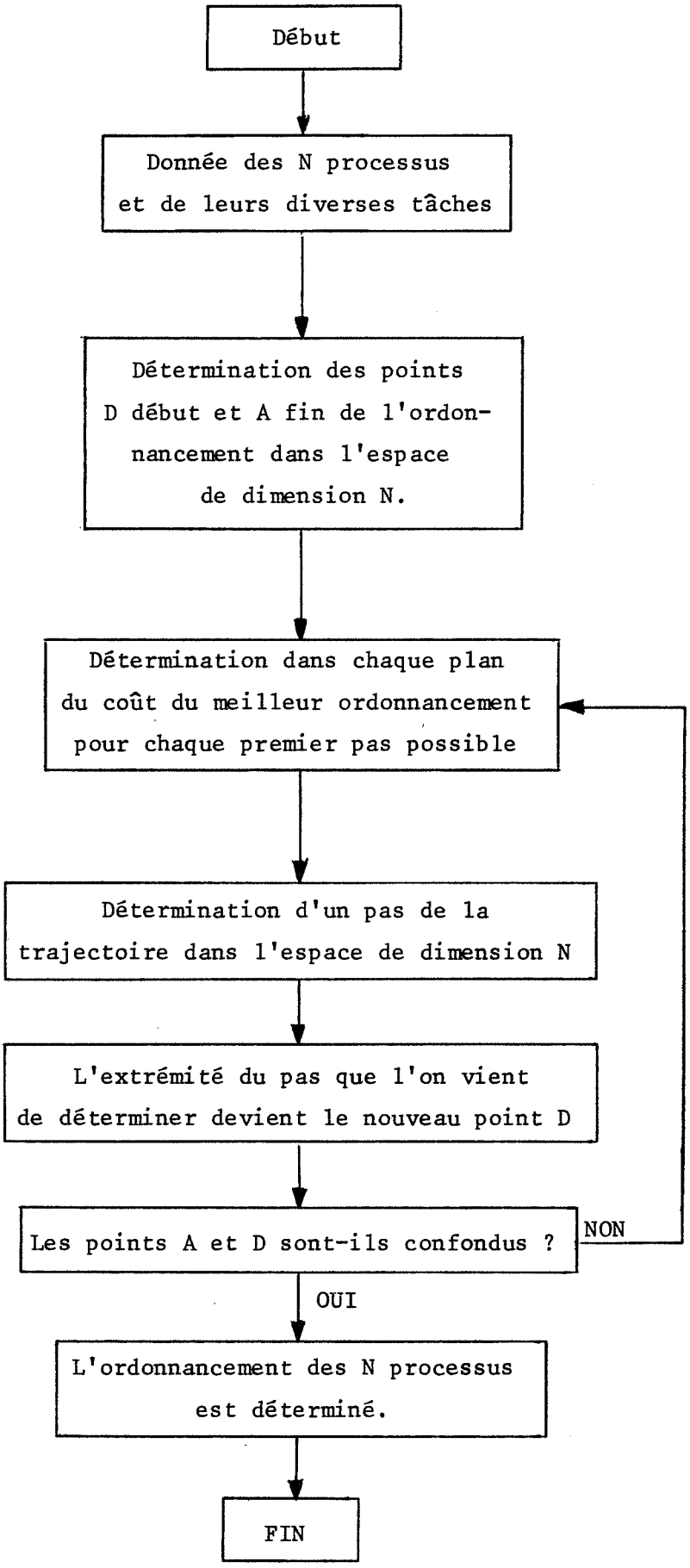
Remarquons que le problème aurait pu être résolu aussi simplement si, au lieu de construire la trajectoire pas par pas à partir du point D, on l'avait construite de façon "rétrograde" à partir du point A.

C'est d'ailleurs de cette façon que procède le "système CYBCO" décrit au chapitre 3.



Représentation du problème à trois dimensions

Organigramme :



Le nombre de plans d'analyse est égal à $\frac{N(N-1)}{2}$. La mise en oeuvre de cette méthode conduit à un grand nombre d'examens de ces plans, en première approximation un examen est fait dans tous les plans à chaque pas (à chaque unité de temps). D'où l'idée que le temps de résolution pourrait être considérablement réduit si on câblait l'étape d'analyse bidimensionnelle.

Dans un premier temps, on a donc pensé pouvoir représenter dans un calculateur spécialisé tous les plans de projection. Si nous reprenons l'exemple précédent, le nombre de points à matérialiser est : $\frac{50 \times 49}{2} \times (10^3)^2 \neq 1{,}225.10^9$, ce qui reste considérable.

C'est pourquoi, on a été conduit à ne câbler qu'un seul plan. La machine est alors capable d'effectuer n'importe quelle analyse bidimensionnelle à condition qu'on lui fournisse chaque fois les caractéristiques du couple de processus concernés. Le nombre de points à matérialiser n'est plus que 10^6.

On a également envisagé de câbler sur cette machine l'étape de recombinaison des trajectoires planes. Ceci n'a pas été réalisé à l'heure actuelle pour des raisons que nous verrons ultérieurement. Les heuristiques de recombinaison sont donc pour l'instant programmées sur un calculateur couplé à la machine.

Diverses équipes de recherche se sont intéressées à la réalisation d'un système câblé permettant de résoudre les problèmes combinatoires sur la base de cette approche.

Citons d'abord la société CYBCO, constituée en 1972, qui s'inspire des travaux du Docteur SAUVAN pour construire un prototype de machine bidimensionnelle dénommée " optimateur". Cette machine a été installée au Centre de Calcul de l'Ecole des Mines de Saint-Etienne et connectée à un ordinateur Télémécanique T1600 de janvier à novembre 1974. Il a d'abord fallu implémenter un logiciel sur le T1600 permettant de piloter "l'optimateur" et d'effectuer la recombinaison des solutions planes ; par la suite nous avons évalué les performances de ce système sur des exemples numériques.

D'un autre côté, la SNCF s'est intéressée à ce type de machine. En 1971, sous la direction du Docteur SAUVAN, M. GENETE soutient une thèse de Docteur-Ingénieur [3] portant sur l'utilisation d'une telle machine à la régulation du trafic ferroviaire. Depuis cette date, un prototype SNCF dénommé combinateur optimisant a été simulé sur ordinateur. Nous en avons construit un modèle pour en évaluer les performances.

Avant d'étudier ces deux réalisations, il est indispensable d'examiner leur origine commune, c'est-à-dire la méthode SAUVAN [8] et le prototype qui en a dérivé. Ce prototype, construit en 1966 par la SNECMA a été implanté à la SNCF en 1970. Il a été abandonné par la suite à cause de ses limitations, mais il fut à l'origine des recherches ultérieures.

2 - 3. DETERMINATION DES TRAJECTOIRES PLANES OU ANALYSE BIDIMENSIONNELLE PAR LA METHODE SAUVAN :

2-3-1- Première méthode SAUVAN de l'analyse bidimensionnelle :

La machine comporte un plan d'analyse constitué de cellules identiques qui propagent un signal. Examinons le mode de fonctionnement de cette machine.

. Fonctionnement de la cellule de base :

Les cellules sont disposées sur une grille représentant le plan. Chaque cellule est connectée en entrée et en sortie à toutes ses voisines. Chaque cellule possède donc 8 entrées et 8 sorties. La progression des signaux est rythmée par une horloge qui fait travailler répétitivement toutes les cellules sur 2 temps :

- 1er temps :

Chaque cellule qui n'a pas encore vu de signal examine les sorties de toutes ses voisines. Si l'une des sorties est un signal, elle passe à l'exécution du 2ème temps. Si toutes les sorties sont nulles, la cellule attend le premier temps suivant.

- 2ème temps :

L'une des sorties de ses voisines étant un signal, la cellule mémorise le signal et le conserve pour tous les temps suivants. Elle coupe ses entrées et n'examinera plus les sorties de ses voisines. Elle émet un signal de sortie vers toutes ses voisines.

De cette façon, un signal émis par une ou plusieurs cellules peut se propager progressivement à toutes les cellules du plan d'analyse en ne passant qu'une fois par chacune d'elles.

Le rôle de cette propagation est de déterminer quel est le plus court chemin pour aller d'une ou plusieurs cellules représentant la situation de départ vers une ou plusieurs cellules représentant la situation but, compte tenu d'un certain nombre de contraintes sur la propagation. Nous avons donc trois états particuliers de cellules à définir :

- la cellule "contrainte",
- la cellule "départ",
- la cellule "but".

. Les différents états de la cellule :

1 - Cellule "contrainte" (ou obstacle).

C'est une cellule qui n'examine pas les sorties de ses voisines et ne se laisse pas traverser par un signal. Un groupe de telles cellules définit un obstacle qui doit être contourné par le signal au cours de la propagation. On pourrait aussi concevoir des cellules ne formant pas un obstacle infranchissable pour le signal comme celles-ci, mais le retardant en nécessitant plusieurs cycles d'horloge avant de le mémoriser.

2 - Cellule "départ".

C'est une cellule qui a mémorisé le signal avant de lancer l'horloge, c'est-à-dire avant de commencer la résolution du problème.

3 - Cellule "but".

Le principe de cette cellule est de bloquer le système sur le premier temps d'horloge dès qu'elle est atteinte par le signal de façon à extraire les coordonnées de la cellule précédente qui vient d'atteindre la situation d'arrivée par sa sortie.

. Résolution d'un problème par cette méthode :

Reprenons le problème exposé en 2-2-1.

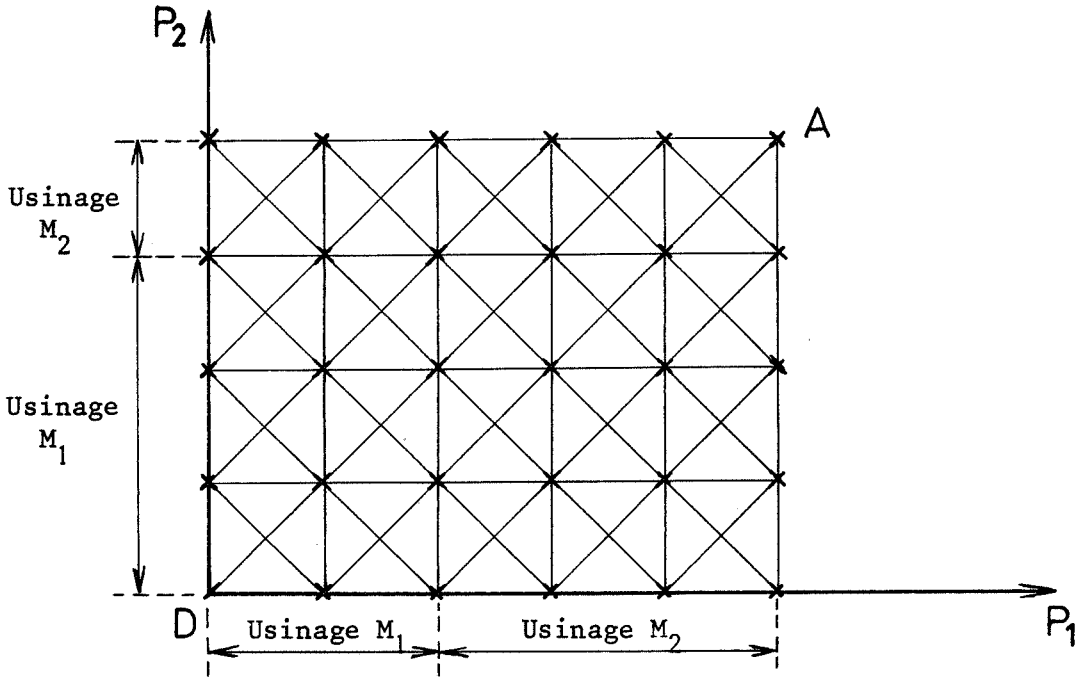


FIGURE 1

Au point D, origine, se trouve une cellule "départ" et au point A, fin du problème, une cellule "but". Supposons qu'il existe aussi une cellule à tous les sommets des carrés marqués d'une croix.

Comme une cellule peut émettre ou recevoir un signal de ses 8 voisines, le signal peut se propager le long de chaque côté et de chaque diagonale des carrés (figure 1). Cependant, pour tenir compte des contraintes en 2-2-1, certaines cellules doivent être des cellules obstacles et d'autres ne peuvent envoyer des signaux à toutes leurs voisines. Sur la figure 2, nous n'avons représenté que les côtés et les diagonales des carrés unitaires susceptibles de transmettre le signal, la règle étant qu'aucun signal ne doit passer à l'intérieur des contraintes représentées sur la figure de 2-2-1.

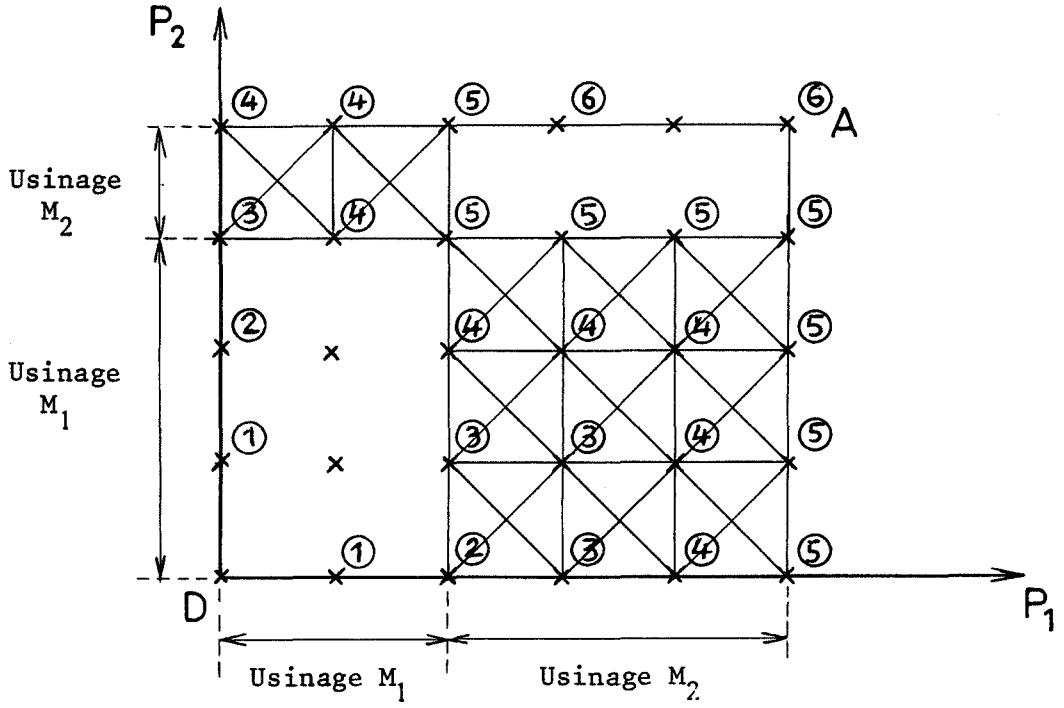


FIGURE 2

Avant le premier cycle d'horloge, le signal est présent dans la cellule "départ" D. Après un cycle d'horloge, toutes les cellules voisines pouvant recevoir le signal sont activées. Dans l'exemple présenté ici, elles sont au nombre de deux, ce sont celles marquées d'un ① . A la fin du second cycle, ce sont celles marquées d'un ② qui sont activées, et ainsi de suite. Au bout de 6 cycles, la propagation s'arrête puisque la cellule "but" située en A a été atteinte.

En remarquant qu'un cycle d'horloge correspond à ce que nous avons appelé un pas, nous constatons que cette méthode nous a permis de trouver le temps minimal nécessaire pour fabriquer les deux pièces du problème d'ordonnancement défini en 2-2-1.

Plus généralement, appelons pour chaque pas :

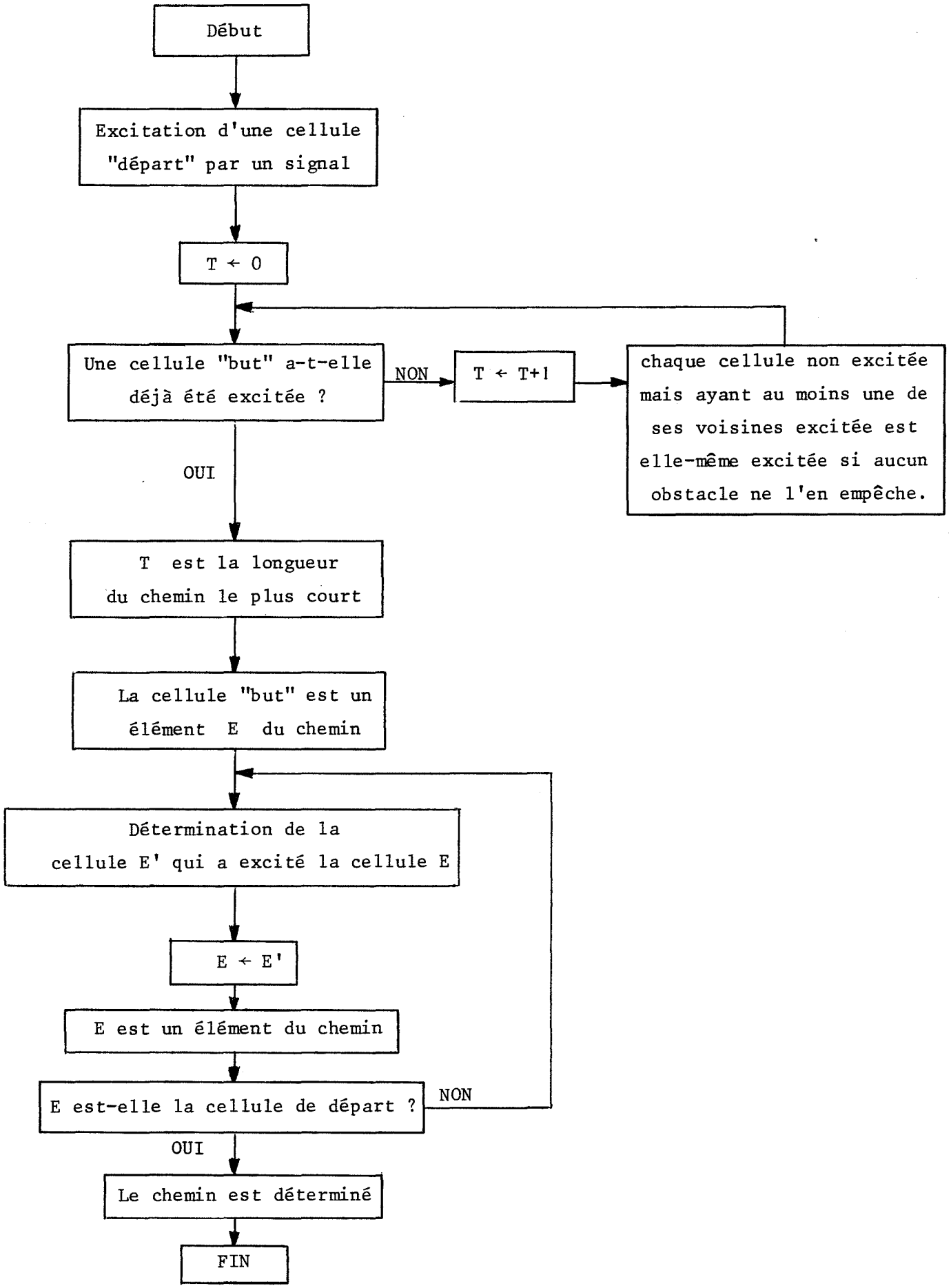
- "dx" la progression du signal en abscisse,
- "dy" la progression du signal en ordonnée.

Remarquons que $dx \in \{0,1\}$ et $dy \in \{0,1\}$ mais que dx et dy ne sont pas simultanément nulles car on aurait alors un point stationnaire sur la trajectoire. Chaque pas a pour longueur 1.

Si, après T cycles d'horloge (ou pas), une cellule "but" est activée, T représente la longueur du chemin le plus court entre une cellule "départ" et une cellule "but".

Nous venons de déterminer le nombre minimal de pas nécessaires pour relier une cellule "but" à une cellule "départ". Pour définir complètement un ordonnancement à deux processus, il nous faut non seulement la longueur du trajet dans le plan mais encore le trajet lui-même. Or, comme chaque cellule retient quelle est, parmi ses prédécesseurs, celle qui l'a excitée initialement, il est facile de retracer en partant du but, le chemin suivi par l'excitation qui a atteint le but en T cycles d'horloge. On obtient ainsi le trajet de longueur minimale joignant un départ à un but.

Organigramme :



. Complexité de la machine et temps de calcul :

La machine que l'on vient de décrire peut travailler sur une grille de $\rho x \rho$ points. Elle nécessite alors ρ^2 cellules. Le temps de calcul θ du plus court chemin est de l'ordre de ρ : $\theta = \sigma(\rho)$. C'est ce qui fait l'avantage de cette méthode par rapport à une méthode logicielle qui nécessiterait un temps de calcul $\theta' = \sigma(\rho^2)$ mais ceci il est vrai, avec une seule unité de calcul au lieu de ρ^2 unités autonomes.

. Justification de la dénomination de "mémoire active" :

On voit maintenant que chacune des ρ^2 cellules comporte à la fois des possibilités autonomes de stockage d'informations et d'envoi de signaux. D'où l'origine du vocable "mémoire active" fréquemment attribué à ce type de machines.

. Introduction des obstacles et lecture du plus court chemin :

Pour qu'une telle machine puisse traiter différents problèmes de plus court chemin, il faut que toutes ses cellules soient identiques et capables de toutes les fonctions. La machine doit être équipée d'une "logique d'affichage" qui permet de "programmer" les problèmes. Cette logique est calquée sur la structure matricielle de la machine et comporte un système d'adressage en X et Y.

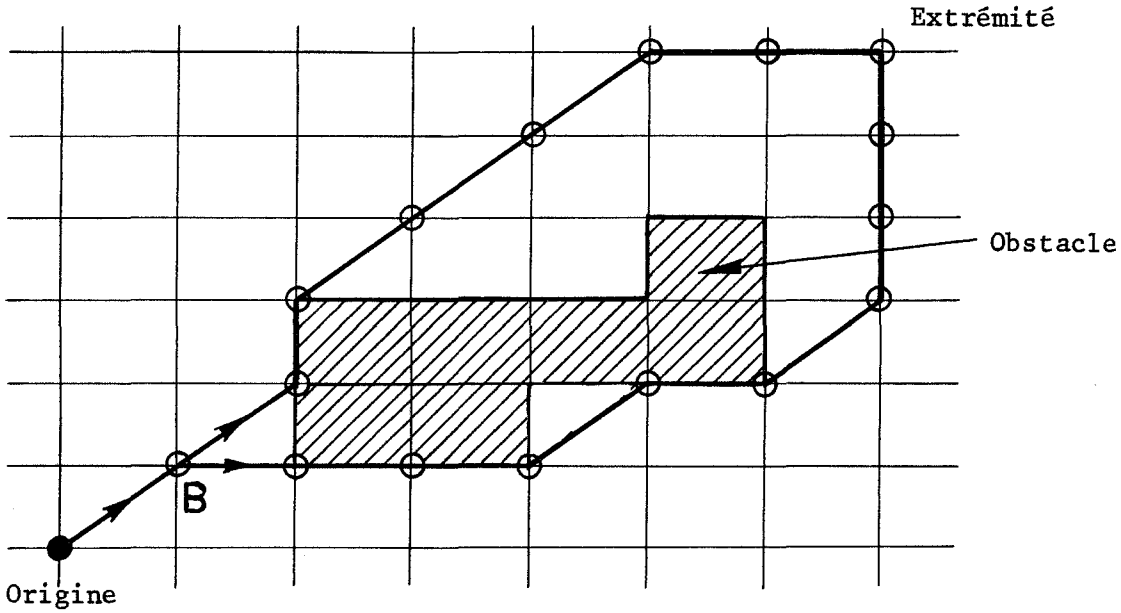
2-3-2- *Etude du prototype SNECMA* :

Devant le coût élevé du réseau cellulaire de la méthode SAUVAN, la SNECMA a préféré réaliser un système plus simple à partir des hypothèses suivantes :

Supposons qu'il y ait un obstacle connexe dans le plan où nous cherchons le chemin le plus court entre un point de départ et un point d'arrivée. Supposons de plus que cet obstacle possède la propriété de convexité diagonale que nous définirons plus loin.

Alors le chemin le plus court passe :

- soit entièrement au-dessus de l'obstacle unique,
- soit entièrement au-dessous de l'obstacle.



Suivons maintenant le fonctionnement du prototype sur la figure ci-dessus. Démarrant à l'origine, le "mobile" s'en va en diagonale jusqu'au point B où il doit prendre la décision de passer au-dessus ou au-dessous de l'obstacle.

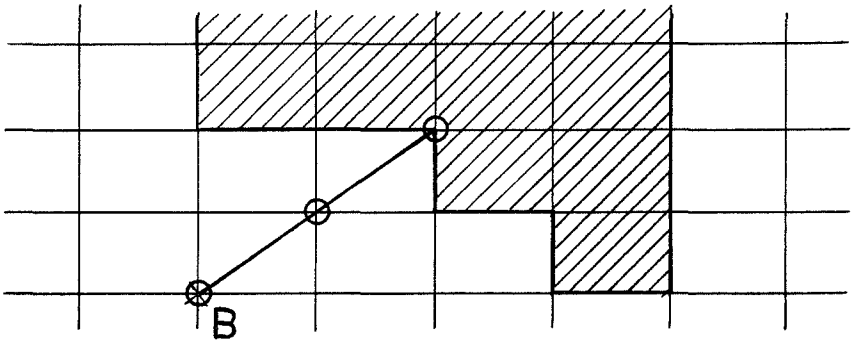
Supposons d'abord que le mobile choisisse le chemin supérieur. On le fait alors avancer en diagonale jusqu'à ce qu'il rencontre l'obstacle (auquel cas on le fait avancer verticalement) ou jusqu'à ce qu'il arrive sur la verticale ou sur l'horizontale de l'extrémité (auquel cas on le fait aller droit à l'extrémité). On trouve alors la longueur du chemin : 8.

De même, on examine le chemin inférieur, et on trouve la longueur 10 dans l'exemple de la figure.

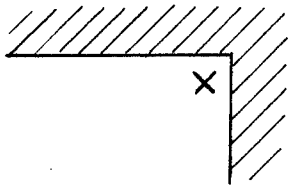
On en déduit que le chemin le plus court passe au-dessus de l'obstacle et a pour longueur 8.

La propriété de convexité diagonale empêche l'arrivée dans une impasse.

Considérons, par exemple, l'obstacle suivant :



La continuation en diagonale à partir de B amène le mobile à un endroit où il est bloqué, car tout retour en arrière est impossible. L'hypothèse de convexité diagonale supprime cette éventualité en interdisant à un obstacle d'avoir un point anguleux rentrant du type de X dans la figure ci-dessous.

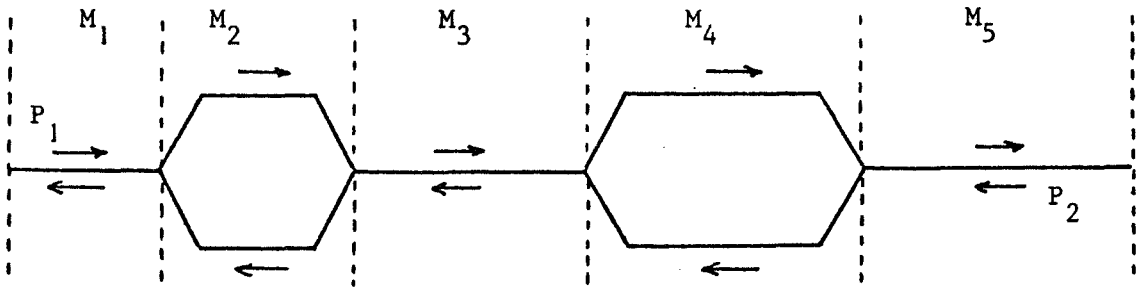


Le prototype mémoire active de la SNECMA résoud, moyennant les restrictions indiquées, le calcul de la longueur du chemin le plus court dans un plan en un temps proportionnel à cette longueur. En réalité, ce prototype peut traiter des problèmes de 6 variables au maximum. Il est composé de 15 centres comme celui décrit ci-dessus, fonctionnant en parallèle et permettant d'étudier simultanément toutes les combinaisons 2 à 2 des 6 variables.

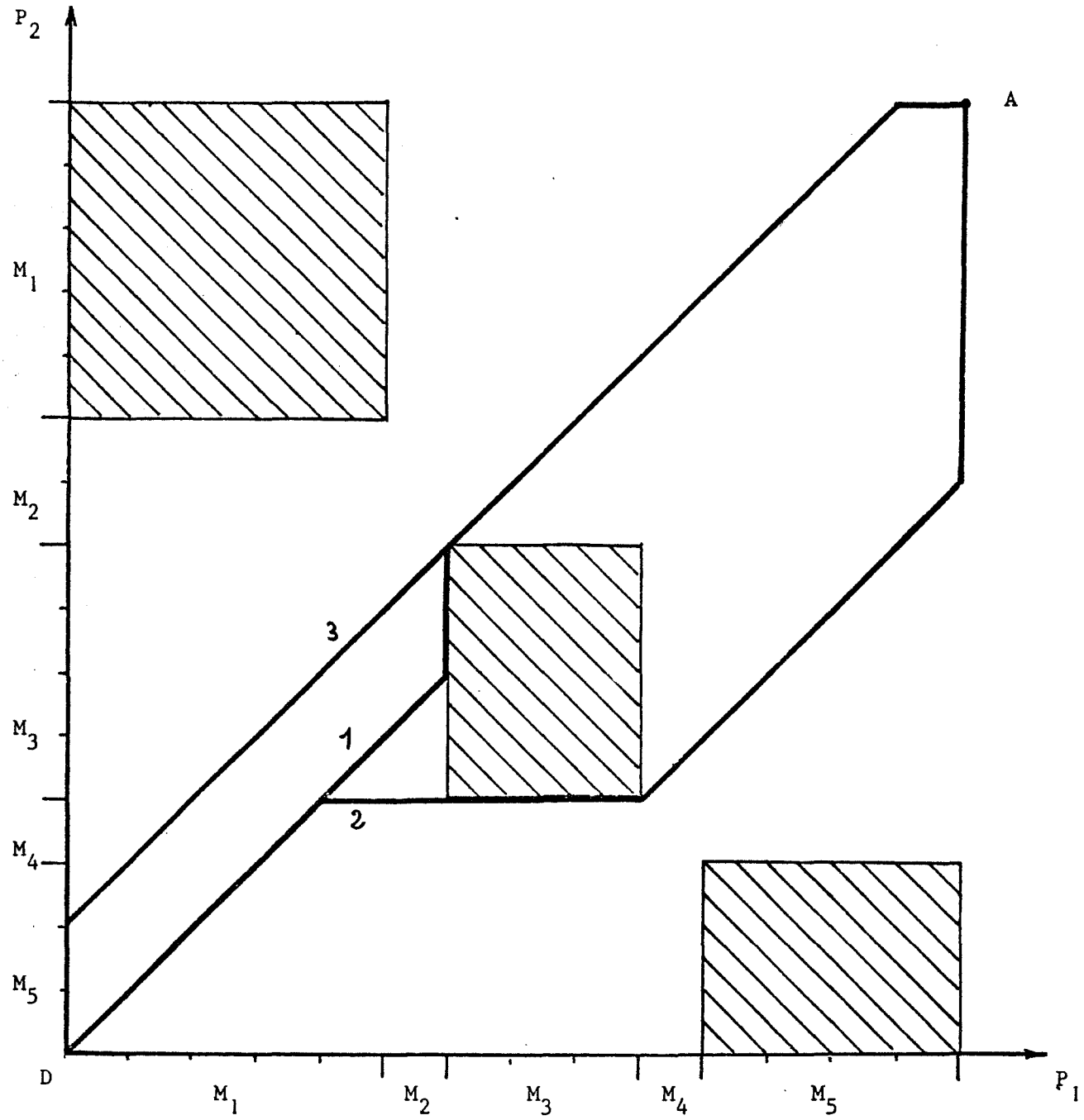
Exemple de calcul :

La méthode précédente va être illustrée par un exemple de régulation de trafic ferroviaire [9].

On considère une voie unique avec évitements M_2 et M_4 (fig a), intercalés sur trois segments de parcours M_1, M_3 et M_5, et on suppose que, pour des raisons de longueur, de tracé, de stationnements éventuels, les trains P_1 et P_2 qui les parcourent ont des marches différentes.



a - Schéma de la ligne



b - Plan d'analyse

P_1 suit la ligne dans le sens M_1, M_2, M_3, M_4, M_5. Les temps de parcours sont respectivement 5, 1, 3, 1, 4. P_2 la suit dans le sens inverse avec des temps 3, 1, 4, 2, 5.

Il est possible, comme nous l'avons vu, figure b, de représenter ces marches sur les axes d'un repère orthonormé. Les contraintes d'interdiction de cohabitation des deux trains sur les segments M_1, M_3 et M_5 conduisent à construire trois obstacles rectangulaires dans le plan d'analyse.

On suppose que le critère à optimiser est le temps séparant le premier départ de la dernière arrivée. En terme d'analyse bidimensionnelle, cela revient à trouver le chemin le plus court reliant les points D et A.

Il est alors aisé d'effectuer la recherche de la trajectoire optimale. Il y a deux manières d'aller de D à A qui correspondent de toute évidence aux croisements en M_2 et M_4.

Dans le premier cas (croisement en M_2 trajectoire 1) la longueur est de 16 ; dans le deuxième (croisement en M_4 trajectoire 2) elle est de 20.

Cet exemple simple montre que la méthode proposée permet de trouver une trajectoire optimale et de connaître sa longueur. Il existe d'autres trajectoires optimales ; au lieu d'immobiliser P_1 en M_2, on peut le faire stationner pendant la même durée à son point de départ (trajectoire 3).

3 - ETUDE DU CALCULATEUR - CYBCO -

Le calculateur CYBCO, ou optimateur, a été conçu pour traiter des problèmes d'ordonnancement en temps réel. Un prototype a été installé en janvier 1974 dans le Centre de Calcul de l'Ecole Nationale Supérieure des Mines de Saint-Etienne, il a été couplé à un ordinateur Télémécanique T1600 pour être testé sur des problèmes concrets.

Le prototype de la SNECMA dont le principe était de faire progresser des signaux dans toutes les directions d'un réseau formé de cellules identiques ne permettait pas de traiter des problèmes dépassant 6 processus à ordonnancer. Il apparaissait nécessaire de concevoir une machine plus performante, c'est ce qu'a essayé de faire CYBCO avec l'optimateur.

3 - 1. ALGORITHME CYBCO DE L'OPTIMATEUR :

Pour des raisons de coût et d'efficacité, un algorithme d'analyse bidimensionnelle ne doit être câblé que s'il présente les caractéristiques suivantes :

- nombre peu élevé de cellules nécessaires à l'analyse bidimensionnelle,
- obtention très rapide des caractéristiques des trajectoires planes utilisées pour la recombinaison.

L'algorithme de l'optimateur a été inventé par Jean Ledieu [10] et adapté par Philippe Eschenbrenner [11]. Nous verrons en 3-1-1 que son principe consiste à calculer les retards minimaux dus aux obstacles dans chaque plan de projection grâce à la déformation d'un front d'onde.

Nous montrerons que dans l'algorithme CYBCO, ce front d'onde est simplement simulé, ce qui permet de réduire considérablement le nombre de cellules nécessaires et ainsi de diminuer le coût de réalisation du calculateur.

3-1-1- *Déformation plane du front d'onde* :

Comme nous l'avons vu en 2-3, la longueur du chemin minimal dans un plan entre deux points D et A est égale au nombre de cycles d'horloge nécessaires pour qu'un signal issu de D parvienne jusqu'en A. Supposons d'abord qu'il n'y a pas de contraintes dans le plan, c'est-à-dire que toutes les cellules sont des cellules actives : cas de la figure 1.

A l'instant 0, le signal a excité la cellule "départ" D,

- à l'instant 1, le signal a excité toutes les cellules voisines, notées 1, de celles précédemment excitées,
- à l'instant 2, le signal a excité les cellules, notées 2, voisines de celles précédemment excitées, et ainsi de suite jusqu'à ce qu'une cellule "but" soit excitée.

Nous appellerons "front d'onde" à l'instant T, l'ensemble des cellules qui ont été excitées au cours de cet instant T. En l'absence de toute contrainte, ce front d'onde est carré, quel que soit l'instant T considéré (figure 1), nous l'appellerons "front d'onde non perturbé" (noté NP).

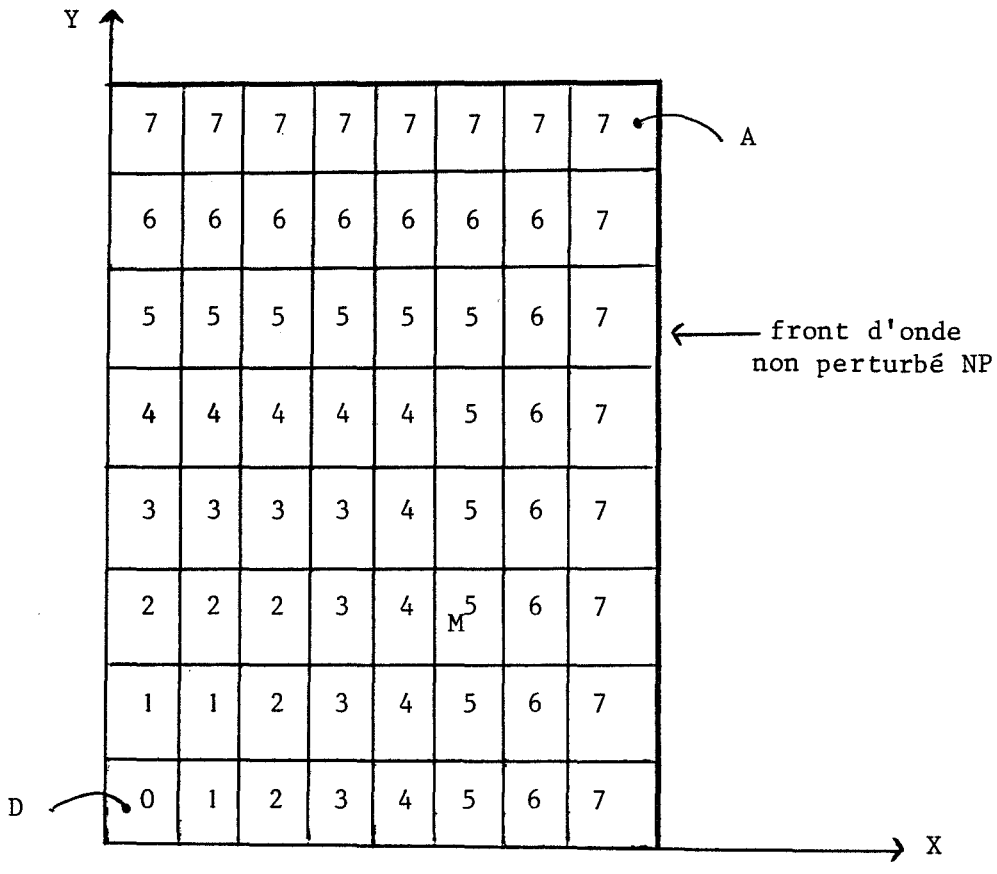


Figure 1 : déplacement d'un front d'onde non perturbé NP.

Si maintenant nous introduisons une contrainte dans le plan d'analyse, c'est-à-dire des cellules obstacles (figure 2), la propagation de l'onde n'est plus possible en certains points du réseau, et le front d'onde résultant présente une allure différente dite perturbée P.

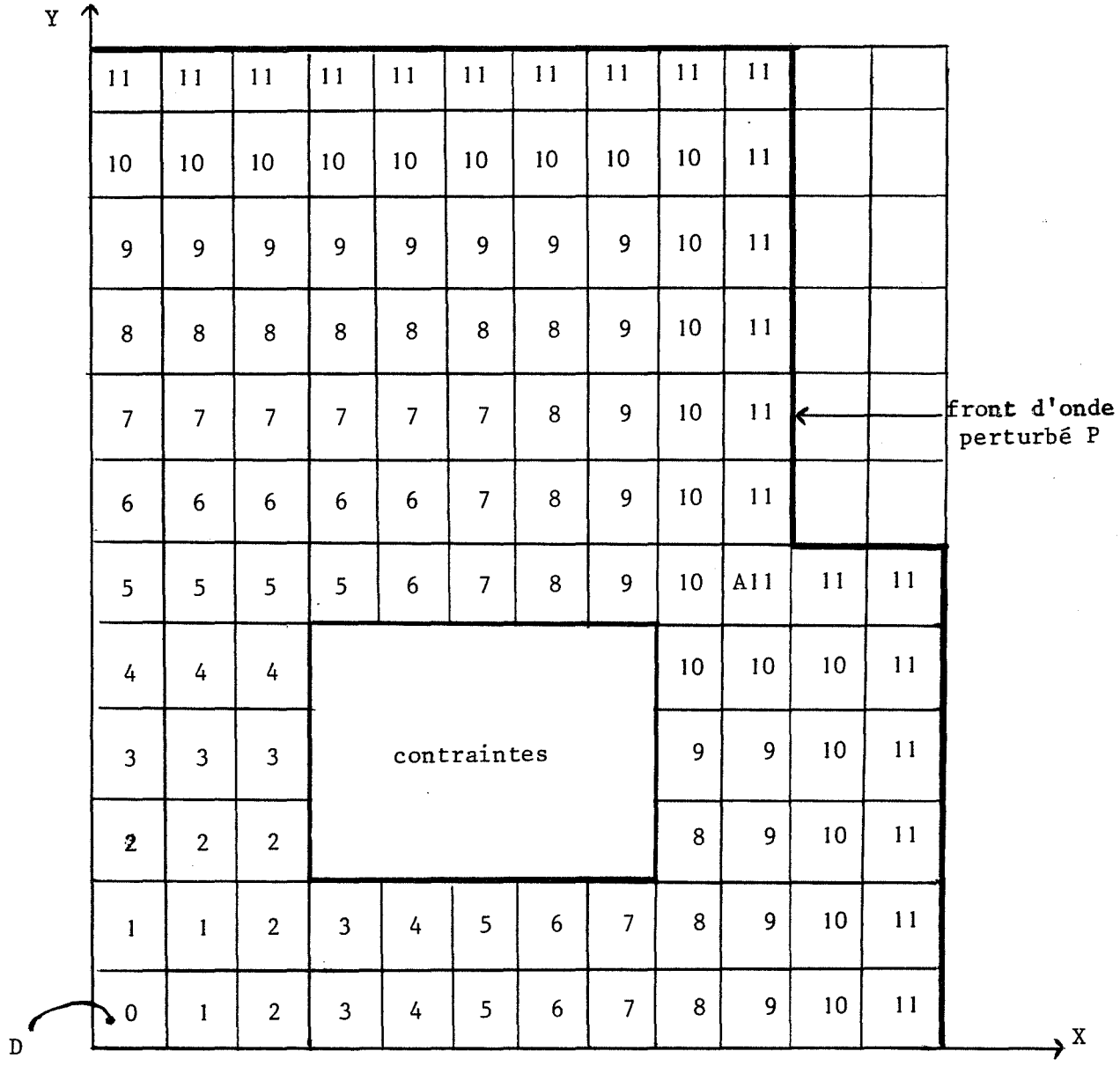


Figure 2 : perturbation du front d'onde par une contrainte.

Nous avons représenté sur la figure 3, simultanément le front d'onde non perturbé et le front d'onde perturbé introduit par la contrainte.

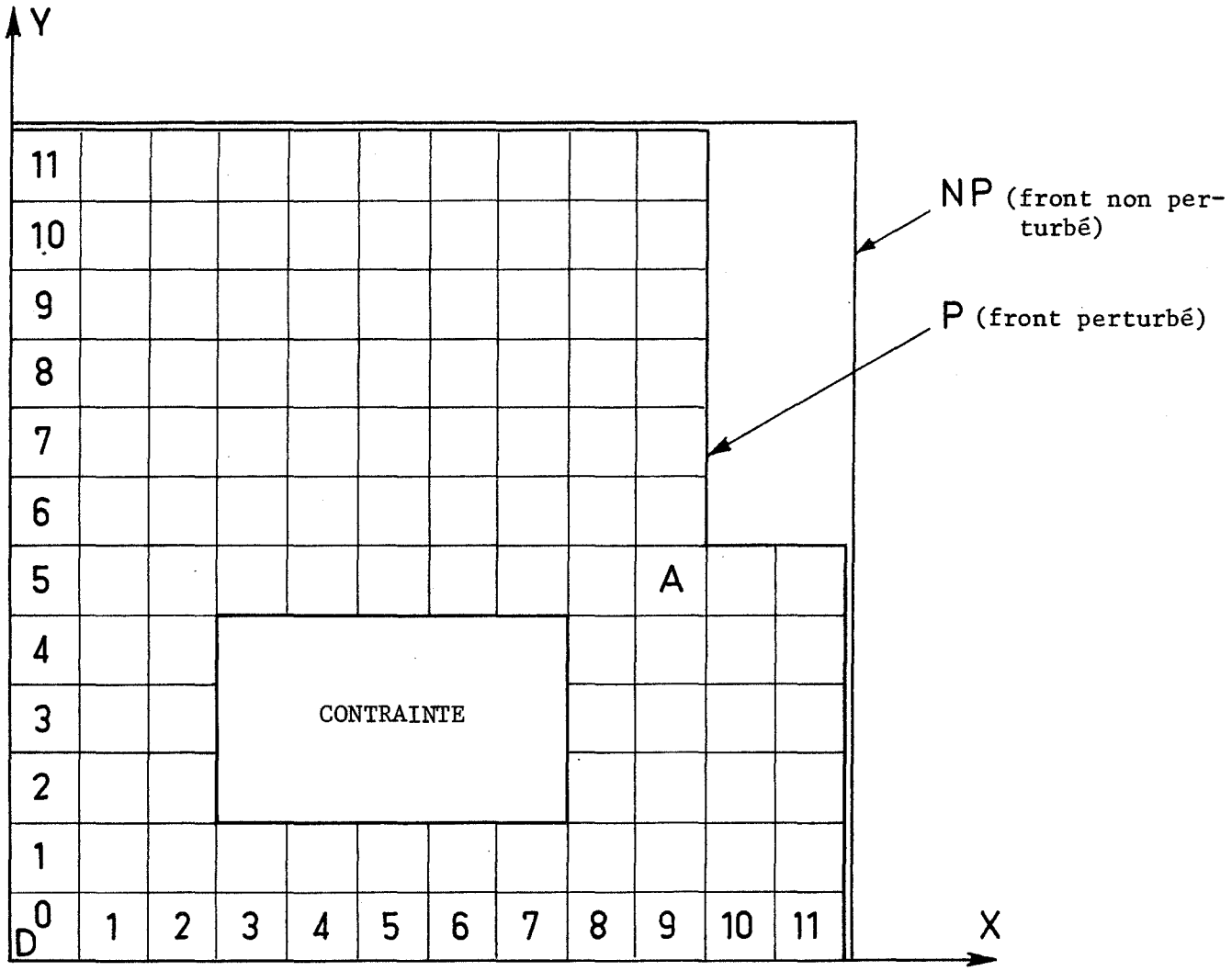


Figure 3 : fronts d'onde perturbé et non perturbé.

L'écart, en cycles d'horloge, entre ces deux fronts d'onde représente le retard dû à la contrainte et est à la base du calcul des longueurs des trajectoires optimales. Avant d'aborder ce calcul, montrons comment déterminer graphiquement le front d'onde lorsqu'il rencontre une ou plusieurs contraintes.

3-1-2- *Détermination du front d'onde perturbé* :

La détermination du front d'onde perturbé se fait graphiquement dans le quadrant opposé à celui de l'analyse. Appelons H et B les points haut et bas de la contrainte, ces points définissent la bande minimale parallèle à la diagonale principale telle que la contrainte y soit incluse en totalité (figure 4).

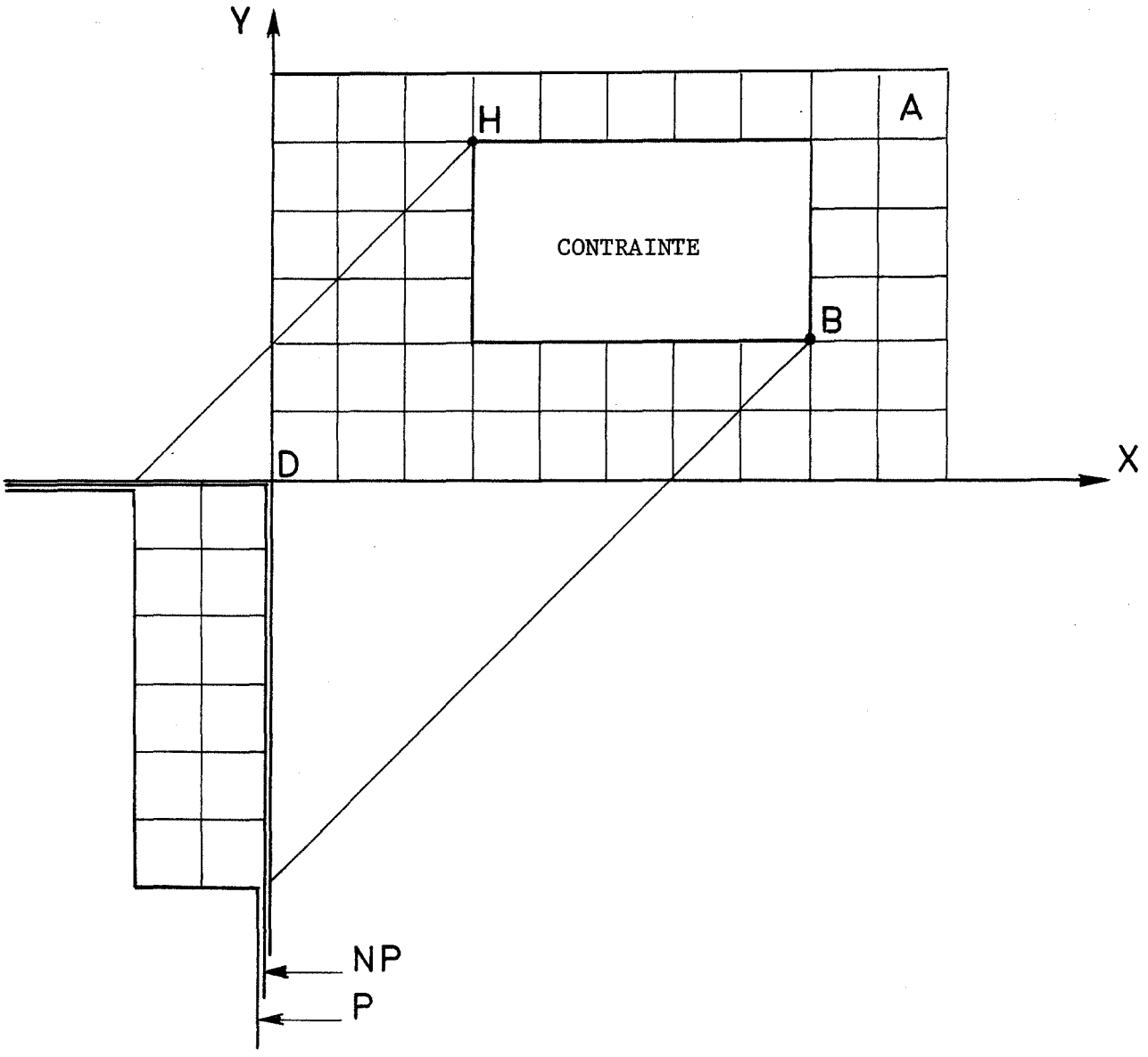


Figure 4 : détermination graphique de la perturbation du front d'onde.

Dans le cas où il n'y a qu'une seule contrainte, il suffit d'effectuer sur les deux demi-axes négatifs la projection diagonale des points haut et bas de la contrainte considérée. Les deux points ainsi obtenus permettent de construire un rectangle de déformation (figure 4) qui est égal à l'écart existant entre les fronts d'onde perturbé et non perturbé (figure 3).

Notons que certaines contraintes peuvent ne pas perturber le front d'onde. Ce sont celles qui sont telles que les projections diagonales de leurs points haut et bas soient situées sur la même droite (figure 5).

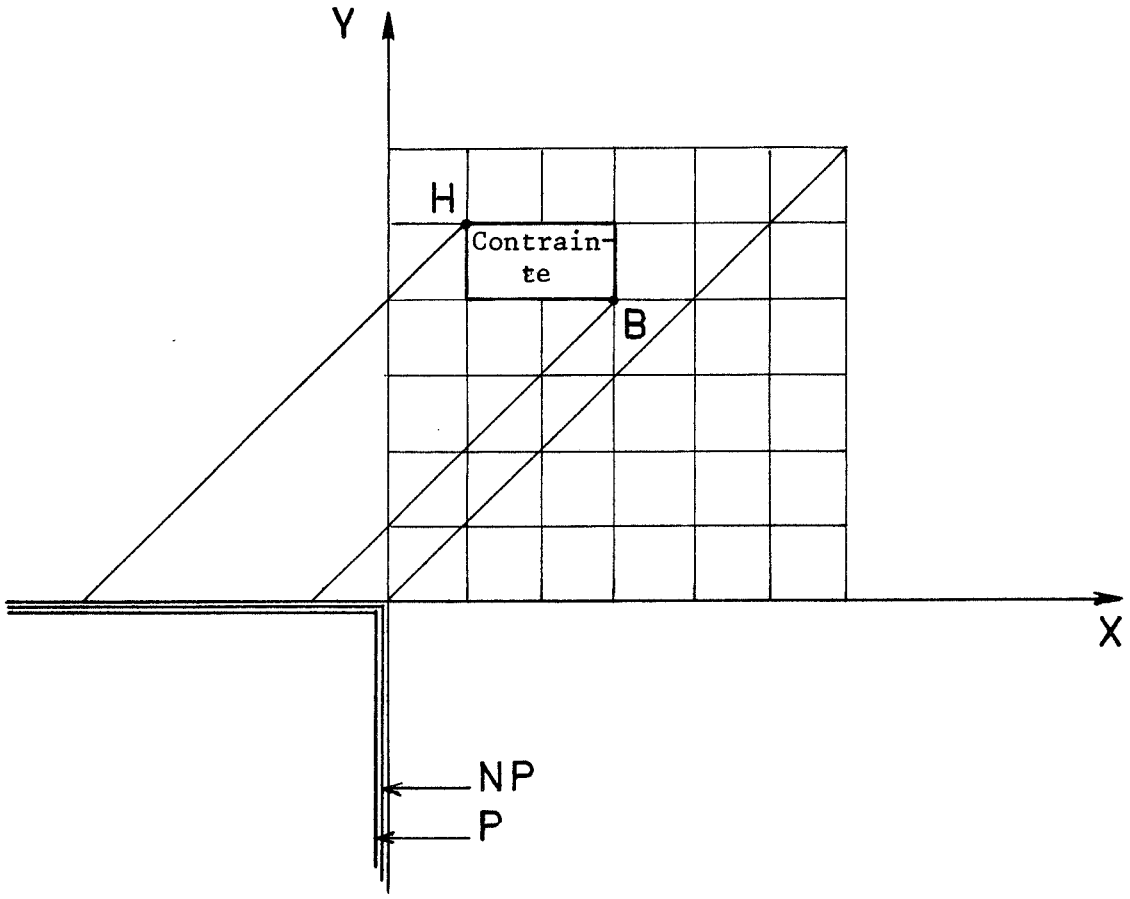


Figure 5 : contrainte non perturbante, les fronts d'onde perturbé et non perturbé sont confondus.

Si le nombre de contraintes du plan est supérieur à 1, il suffit de les projeter l'une après l'autre sur le front d'onde perturbé induit par les précédentes. Aussi, dans le cas de la figure 6, la projection de la contrainte 1 définit une première perturbation. La contrainte 2 est à son tour projetée sur cette perturbation, ce qui détermine un nouveau front d'onde perturbé, et ainsi de suite jusqu'à épuisement des contraintes. Ceci pose le problème de l'ordre dans lequel les contraintes doivent être projetées : cet ordre a en effet une influence sur la forme du front résultant. Ce problème n'est pas abordé ici, il le sera en 3-2.

Dans le cas de figure présenté, l'ordre correct est celui de la numérotation.

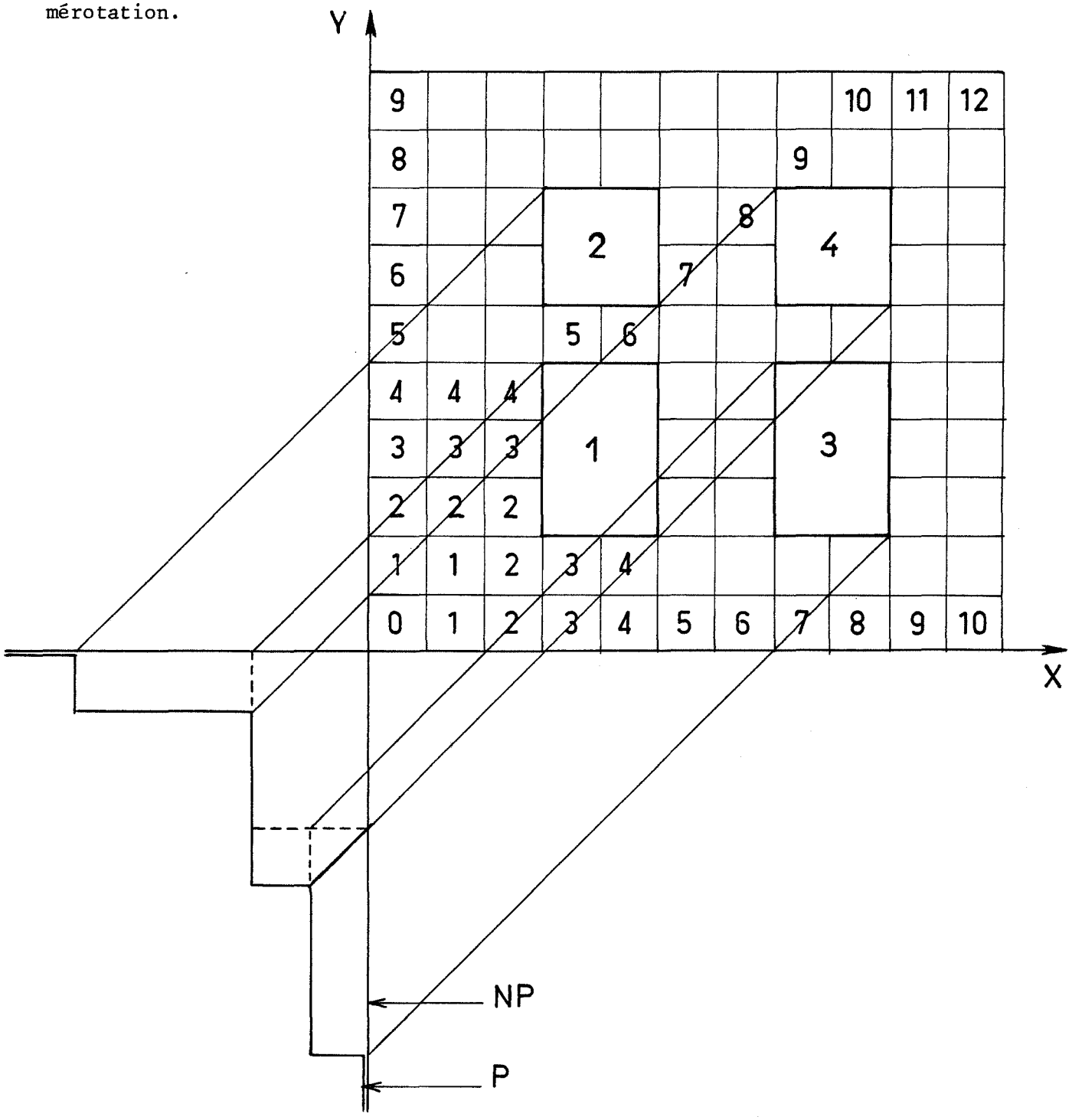


Figure 6 : Perturbation du front d'onde par plusieurs contraintes.

Sur la figure 7 enfin nous avons représenté un cas où la prise en compte d'une contrainte supplémentaire ne modifie pas le front d'onde.

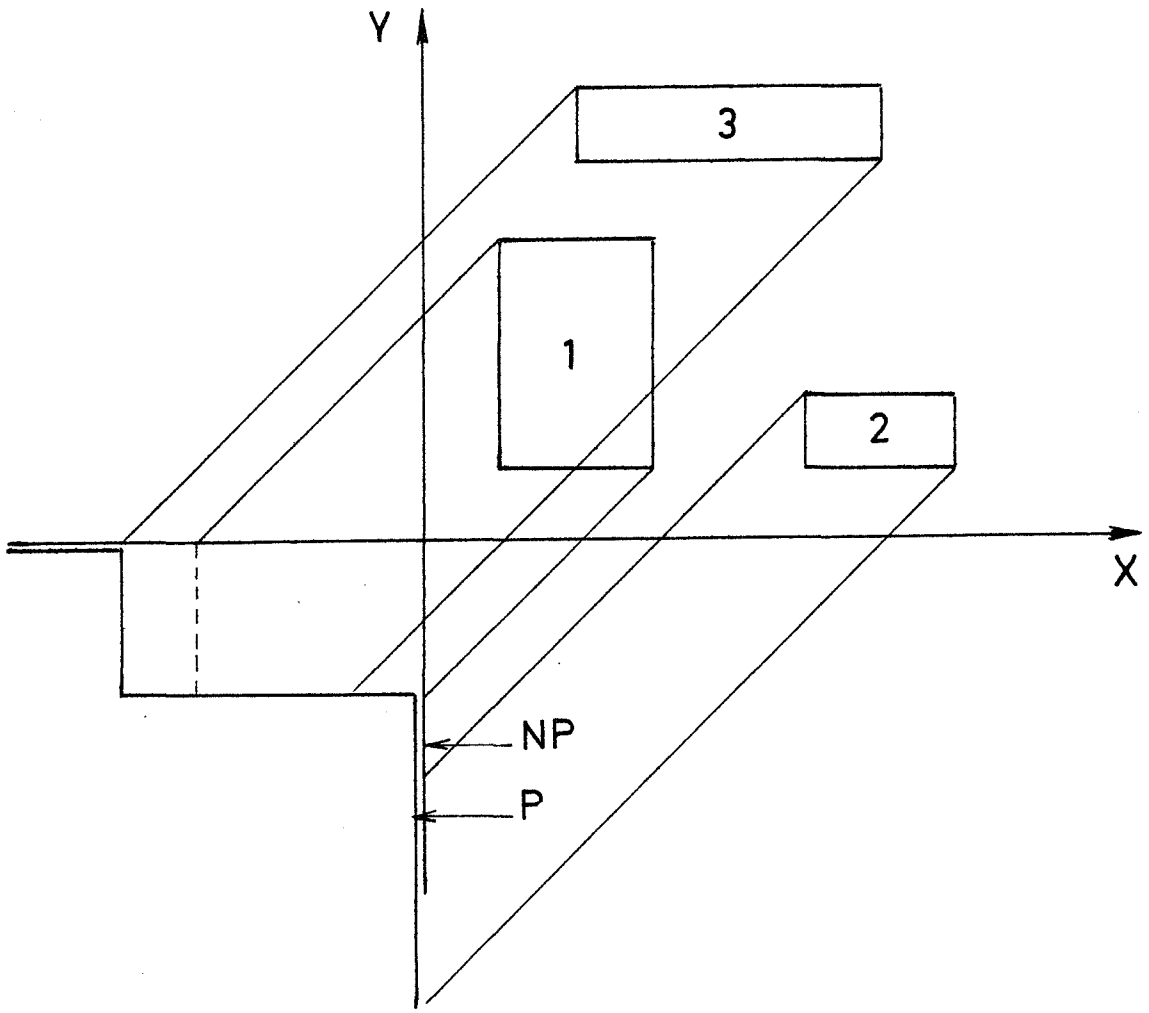


Figure 7 : la contrainte 2 ne modifie pas un front déjà perturbé.

3-1-3- *Calcul des longueurs des trajectoires* :

Considérons la figure 1 représentant le déplacement d'un front d'onde dans un plan non perturbé. Appelons $T_{NP}(M)$ le nombre de cycles d'horloge nécessaires pour atteindre un point M quelconque du plan, de coordonnées x et y, $T_{NP}(M)$ est égal à la plus grande des valeurs x et y :

$$T_{NP}(M) = SUP\ (x,y)$$

A l'aide des figures 3 et 4, nous avons vu comment retrouver graphiquement le front d'onde perturbé. Appelons DELTA (A) le retard au point A du front d'onde perturbé par une ou plusieurs contraintes. Si L est la longueur de la portion de la diagonale passant par A et comprise entre les fronts d'onde perturbé et non perturbé (figure 8), DELTA (A) = $L/\sqrt{2}$

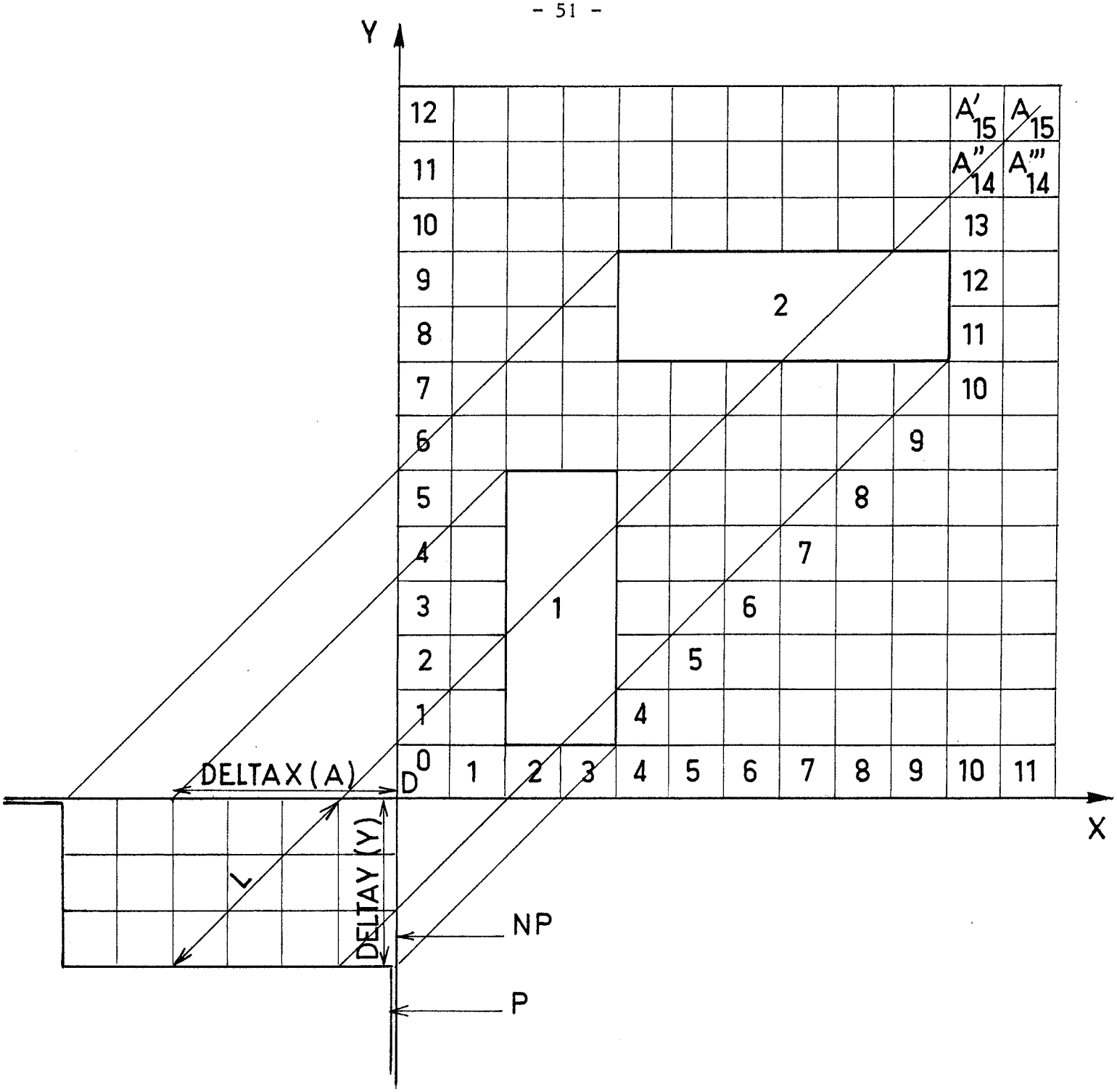


Figure 8 : calcul des retards DELTA(A), DELTAX(A), DELTAY(A).

Posons :

- $T_{NP}(A)$, temps mis par le front d'onde non perturbé pour atteindre A.
- $T_P(A)$, temps mis par le front d'onde perturbé pour atteindre A.

Le retard DELTA(A) s'écrit :

$$\boxed{DELTA(A) = T_P(A) - T_{NP}(A)}$$

Pour trouver $T_P(A)$, temps mis par le front d'onde perturbé pour atteindre A ou longueur du trajet le plus court entre D et A, il suffit donc d'ajouter à la plus grande des coordonnées x_A et y_A de A dans le plan d'analyse, le retard DELTA (A).

$$T_P(A) = T_{NP}(A) + DELTA\ (A)$$

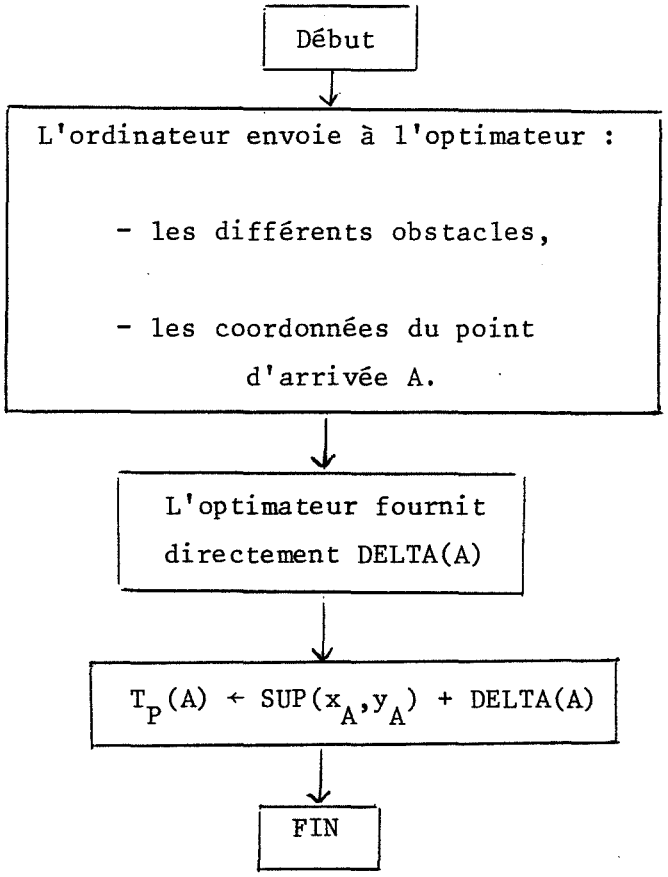
or $T_{NP}(A) = SUP(x_A, y_A)$, donc :

$$T_P(A) = SUP(x_A, y_A) + DELTA(A)$$

Dans le cas de la figure 8, cette longueur est :

$$T_P(A) = SUP(11,12)+3 = 15$$

Organigramme :



Comme nous le verrons plus loin, la mise en oeuvre de la recombinaison exige en outre la connaissance des longueurs des trajectoires aboutissant aux 3 prédécesseurs A', A", A"' du point A.

Les longueurs sont calculées de la même façon que pour le point A :

$T_P(A')$ = SUP(10,12) + 3 = 15
$T_P(A")$ = SUP(10,11) + 3 = 14
$T_P(A"')$ = SUP(11,11) + 3 = 14

Une variante de cette méthode consiste à calculer la longueur du trajet en ajoutant à l'une des coordonnées du point d'arrivée le retard suivant cette coordonnée (figure 8). Si le calcul est effectué suivant l'axe X, appelons DELTAX(A) ce retard :

$T_P(A)$ = x_A + DELTAX(A) = 11+4 = 15
$T_P(A')$ = $x_{A'}$ + DELTAX(A') = 10+5 = 15
$T_P(A")$ = $x_{A"}$ + DELTAX(A") = 10+4 = 14
$T_P(A"')$ = $x_{A"'}$ + DELTAX(A"') = 11+3 = 14

Il est évident que ces calculs peuvent être réalisés de la même manière sur l'axe Y. A titre d'exemple, nous les réaliserons sur l'axe X.

L'utilisation de cette méthode apparaît peu commode et coûteuse dans la mesure où elle nécessite la construction d'une grille plane formée de cellules. L'originalité de l'analyse bidimensionnelle CYBCO consiste à simuler le plan d'analyse en utilisant un nombre très réduit de cellules, on dit qu'on "linéarise le front d'onde".

3-1-4- *Linéarisation du front d'onde* :

Nous avons vu que pour déterminer le front d'onde perturbé, on tenait compte de la position des diagonales supportant les points haut et bas des contraintes et non de la position de ces points sur les diagonales. De même, pour calculer le retard du front d'onde au point d'arrivée, on considère simplement la diagonale du point d'arrivée. Ces remarques ont conduit les réalisateurs de l'optimateur à associer une cellule à chaque diagonale. Le gain, en nombre de cellules, est important ; en effet, si chaque axe est discrétisé en p pas, le nombre de diagonales, donc de cellules nécessaires, est 2p + 1 au lieu de p^2 si on utilisait une grille de cellules. C'est cette simplification apportée à la méthode qu'on appelle la "linéarisation du front d'onde".

Le nombre de coordonnées diagonales qui peuvent intervenir est égal au nombre de diagonales traversant l'intervalle d'étude. Si :

- X est le nombre de diagonales traversant l'axe des X positifs,
- Y est le nombre de diagonales traversant l'axe des Y positifs, le nombre N_{CD} de coordonnées diagonales qui peuvent intervenir est :

$N_{CD} = X+Y+1$, le 1 correspondant à la diagonale principale.

Sur la figure 9, $N_{CD} = 11+12+1 = 24$.

L'analyse bidimensionnelle de ce problème ne nécessite que 24 cellules.

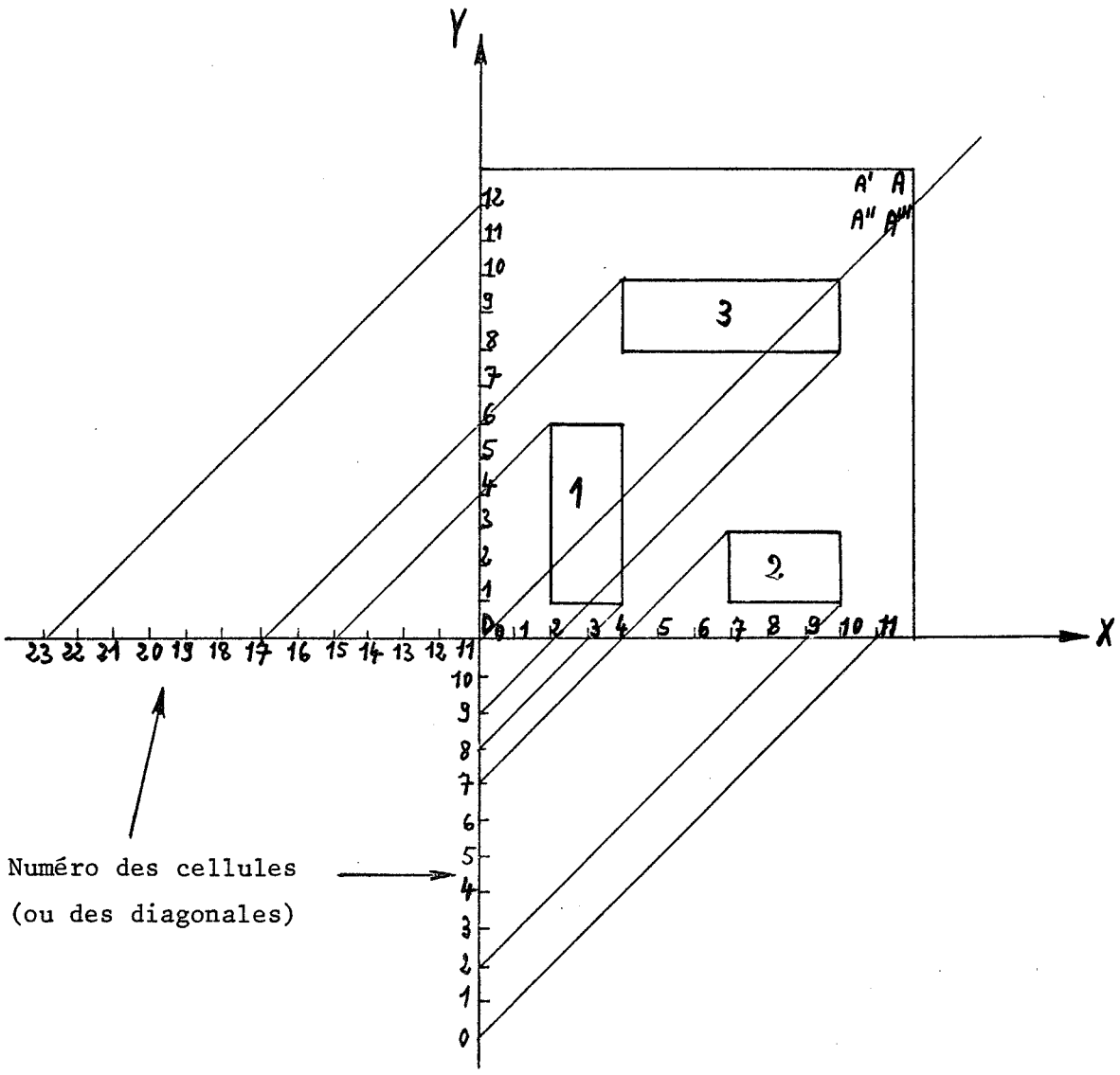


Figure 9 : Affectation des cellules aux diagonales.

Avant tout traitement, le contenu de chacune de ces cellules doit être égal au DELTAX de la diagonale correspondante, pour cela il convient d'initialiser en mettant des 0 dans les cellules numérotées de 0 à 11 (numéro de la diagonale principale) et les nombres 1,2,3,...., 12, respectivement dans les cellules 12 à 23, (figure 10).

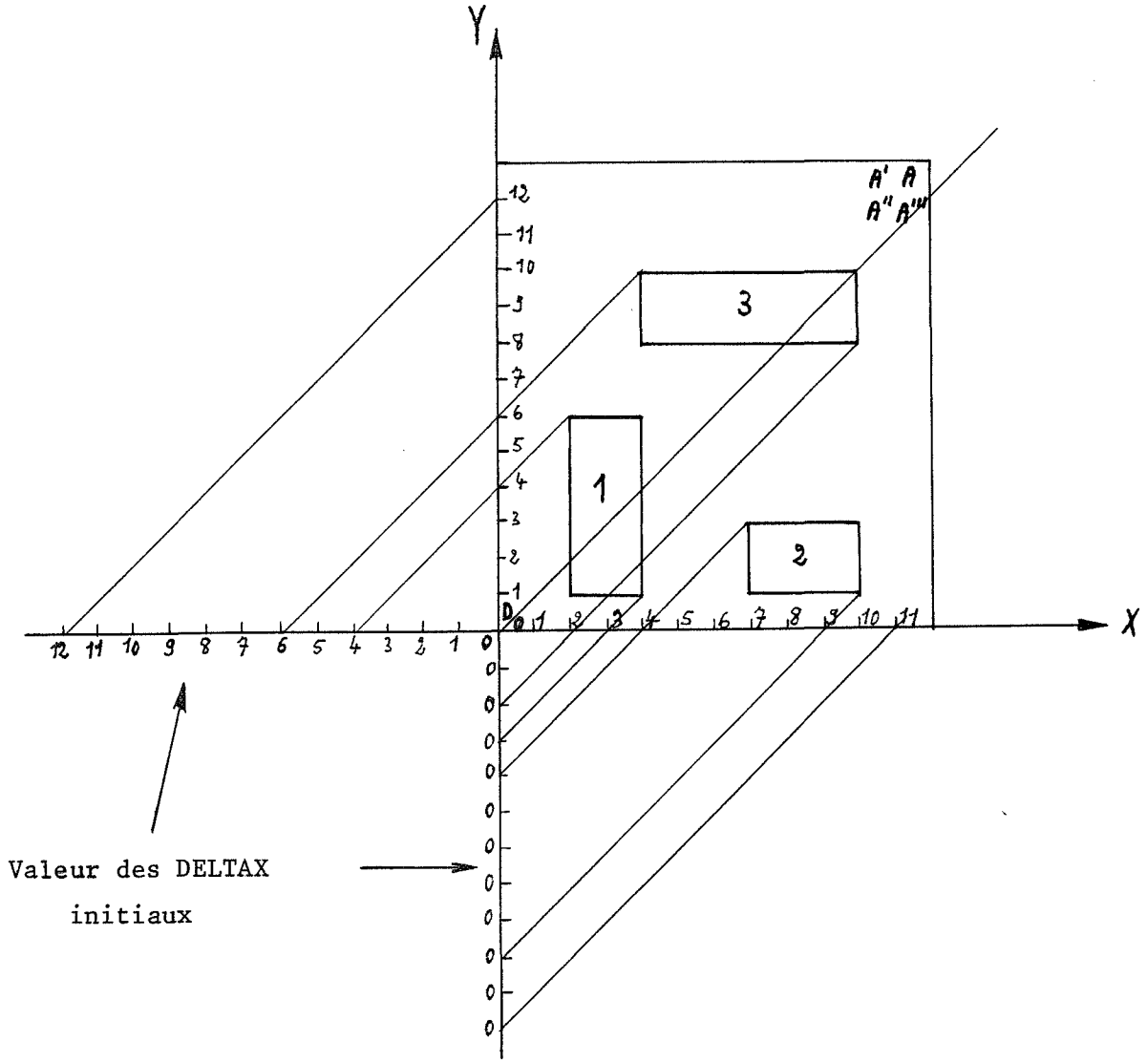


Figure 10 : Initialisation des cellules.

Chaque contrainte est alors caractérisée par ses adresses haute et basse. Pour la contrainte 1, (figure 9), nous avons H = 15 et B = 8. Si on désigne par (X) le contenu de la cellule numéro X, le traitement de cette contrainte peut être décrit de la manière suivante :

1 - Si $X \leq B$, (X) reste inchangé.

2 - Le contenu de la cellule d'adresse B + 1 passe à (B) + 1 ; celui de la cellule d'adresse B + 2 à (B) jusqu'à ce qu'on arrive à la cellule dont le contenu devienne égal à (H).

3 - Le contenu des cellules non encore examinées et d'adresse inférieure à H est rendu égal à (H).

4 - Si $X > H$, (X) reste inchangé.

Les calculs sont effectués dans le tableau I. La contrainte 2 n'a pas été prise en compte puisque nous avons vu qu'elle n'intervenait pas. Le procédé peut encore être simplifié : le calcul n'est plus effectué sur le contenu de la cellule mais sur les incréments c'est-à-dire la différence entre son contenu propre et celui de la cellule qui la précède.

La prise en compte d'un obstacle d'adressesH et B conduit alors aux calculs suivants :

1 - Faire la somme S des contenus des cellules X telles que $B < X \leq H$.

2 - Si $X \leq B$, (X) reste inchangé.

3 - Si $B+1 \leq X \leq B+S$, (X) = 1.

4 - Si $B+S+1 \leq X \leq H$, (X) = 0.

5 - Si $X > H$, (X) reste inchangé.

Le tableau II résume ces calculs.

TABLEAU I

o=o=o=o=o=o=o

N° cellule	Contenu Initial	Traitement contrainte 1	Traitement contrainte 3	Contenu final
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	1	1	1
10	0	2	2	2
11	0	3	3	3
12	1	4	4	4
13	2	4	5	5
14	3	4	6	6
15	4	4	6	6
16	5	5	6	6
17	6	6	6	6
18	7	7	7	7
19	8	8	8	8
20	9	9	9	9
21	10	10	10	10
22	11	11	11	11
23	12	12	12	12

Les longueurs des trajets sont obtenus en ajoutant à l'abscisse des différents points d'arrivée le DELTAX contenu dans les cellules de leur adresse diagonale.

$$T_p(A) = 11 + 4 = 15$$
$$T_p(A') = 10 + 5 = 15$$
$$T_p(A'') = 10 + 4 = 14$$
$$T_p(A''') = 11 + 3 = 14$$

TABLEAU II

o=o=o=o=o=o=o=o=o=o

N° Cellule	Contenu Initial	Incrément Initial	Contrainte 1 Incrément	Contrainte 3 Incrément	Contenu final
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	1	1	1
10	0	0	1	1	2
11	0	0	1	1	3
12	1	1	1	1	4
13	2	1	0	1	5
14	3	1	0	1	6
15	4	1	0	0	6
16	5	1	1	0	6
17	6	1	1	0	6
18	7	1	1	1	7
19	8	1	1	1	8
20	9	1	1	1	9
21	10	1	1	1	10
22	11	1	1	1	11
23	12	1	1	1	12

Dans la deuxième colonne, nous avons fait figurer les contenus initiaux au sens du tableau I ; nous en avons déduit dans la troisième colonne les incréments initiaux. A partir des incréments finaux, (colonne 5), nous obtenons le contenu final, au sens du tableau I, par sommation.

Cette dernière version conduit au même contenu final des cellules que la précédente. Elle présente l'avantage de travailler en base binaire. C'est la raison pour laquelle elle a été choisie pour être câblée sur la machine.

3 - 2. PREPARATION DES DONNEES ET CAS D'UTILISATION DE L'OPTIMATEUR :

3-2-1- Obtention des obstacles :

Les données d'un problème d'ordonnancement sont parfaitement connues lorsque sont fixés pour chacune des N pièces :

- l'ordre de passage sur chaque machine (gamme de fabrication),
- les durées de traitement sur chaque machine.

Avant de commencer les calculs, la première opération consiste à décrire chacun des $\frac{N(N-1)}{2}$ plans de projection.

On considère chaque couple de processus ou variables indépendamment des autres et, compte tenu de leurs gammes, on détermine les coordonnées des obstacles du plan qui correspondent à l'utilisation simultanée d'une même machine.

On détermine les coordonnées spatiales du point d'arrivée. La coordonnée relative à l'axe représentatif d'une pièce est égale à la durée totale de fabrication de cette pièce.

Taille des tables nécessaires pour enregistrer les données dans l'ordinateur :

Tous les obstacles doivent être conservés en mémoire de l'ordinateur qui les envoie ensuite à l'optimateur. Soit $\mathcal{N}$ le nombre total d'obstacles. Chaque obstacle requiert 4 mots (Xmin, Xmax, Ymin, Ymax) ; il faut donc prévoir $4\mathcal{N}$ mots.

Avec N = 40 variables, $K = \frac{N(N-1)}{2} = 780$ plans et avec 3 obstacles en moyenne par plan, $\mathcal{N} = 3\ K = 2340$. Le nombre de mots nécessaires est alors de $4 * 2340 = 9360$ mots.

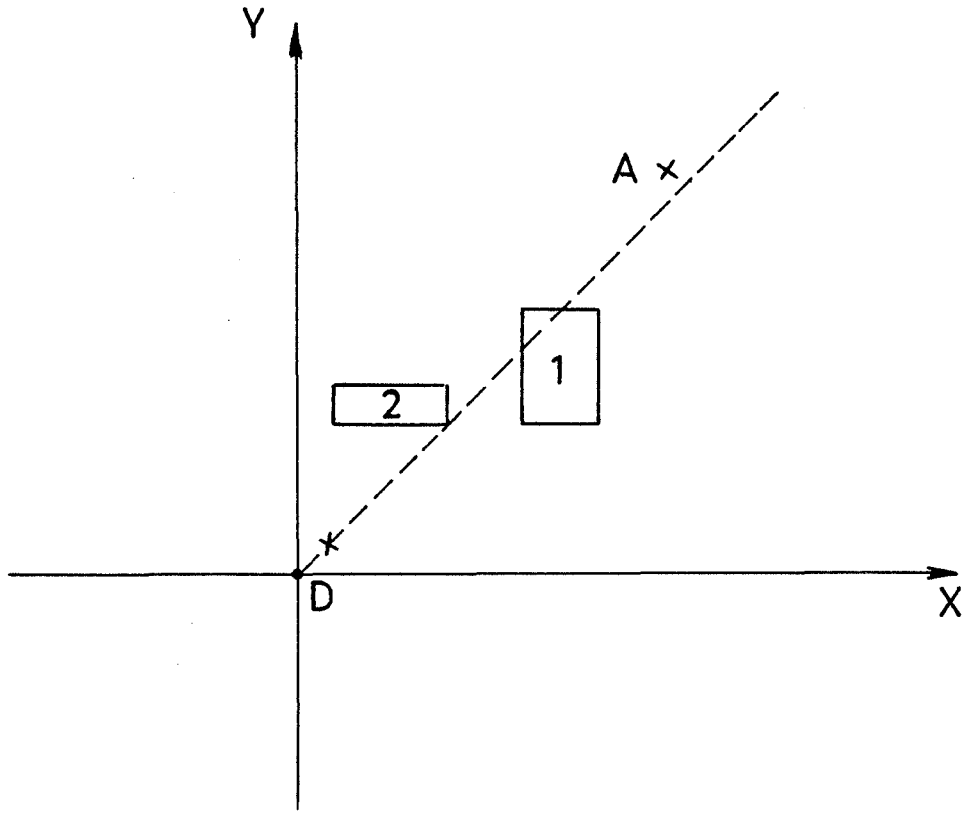
Compte tenu de la taille du programme et du système d'exploitation, le traitement nécessite un T1600 avec 16 K mots de 16 bits.

L'utilisation du disque comme mémoire auxiliaire permettrait, moyennant une augmentation du temps d'exécution, de traiter des masses de données plus volumineuses.

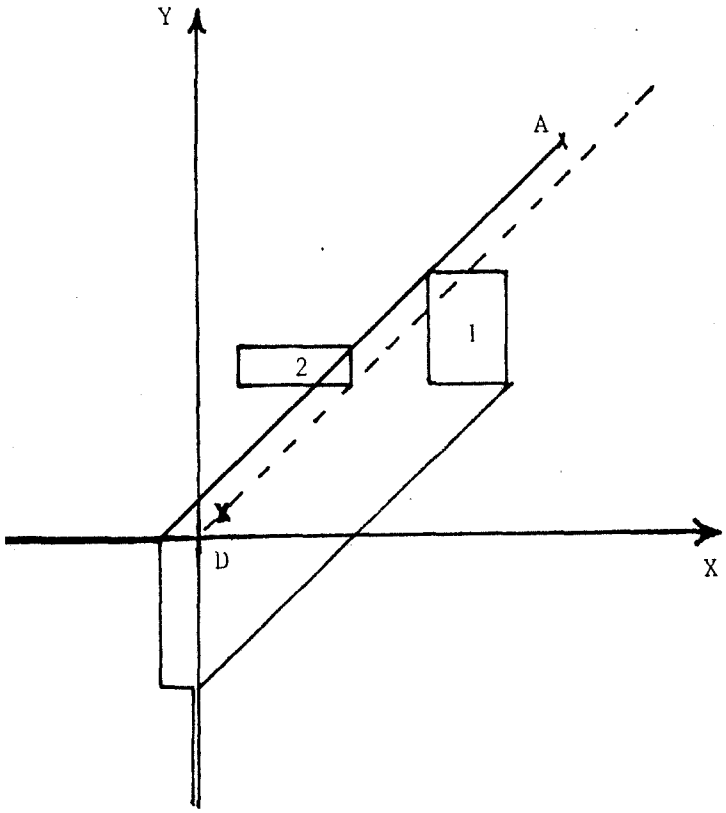
3-2-2- *Ordonnancement des obstacles* :

Comme nous l'avons vu plus haut, pour être traité par l'optimateur, un problème d'ordonnancement à N processus ou variables doit être projeté sur $\frac{N(N-1)}{2}$ plans de projection, les axes de coordonnées de ces plans représentant l'ensemble des processus pris deux à deux. Sur chaque plan de projection on aura donc la projection des points de départ et d'arrivée dans l'espace à N dimensions ainsi que la projection des différentes contraintes.

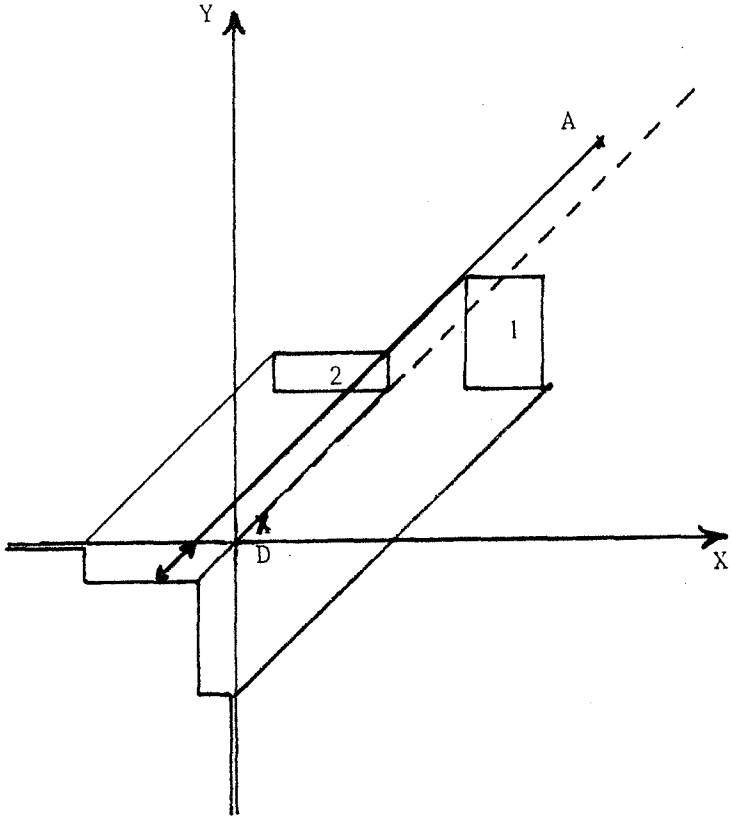
Nous allons dans ce paragraphe, poser le problème de l'ordre dans lequel les obstacles, pour un plan donné, doivent être présentés à l'optimateur CYBCO.



Nous nous donnons donc un plan qui, par exemple, contient 2 contraintes. En appliquant la méthode décrite en 3-1-2, si nous traitons d'abord la contrainte 1 puis la contrainte 2, nous obtiendrons successivement les schémas suivants :



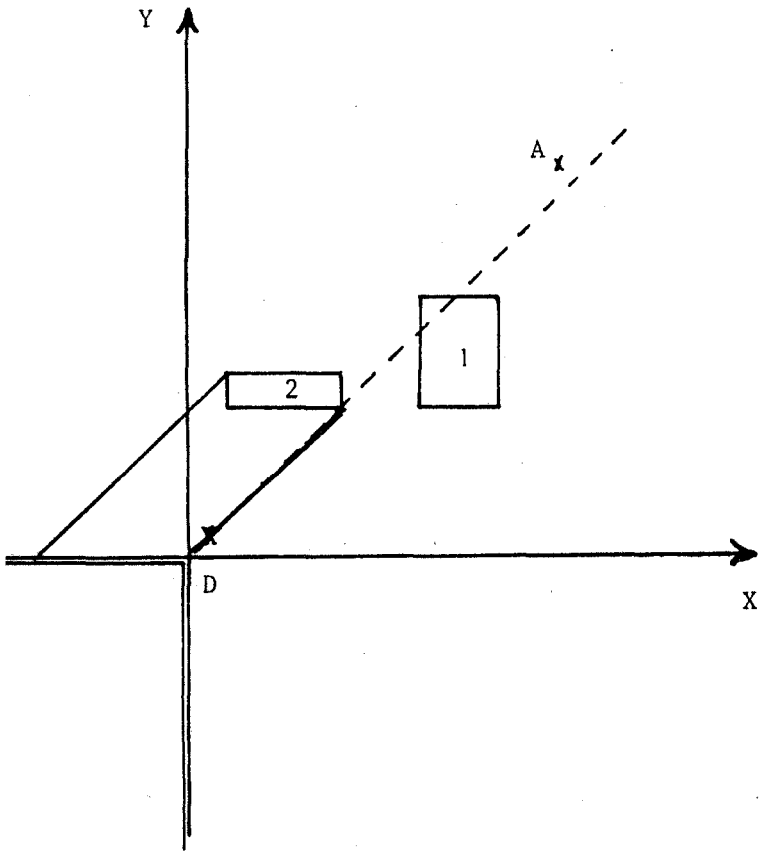
Retard du front d'onde après traitement de la contrainte 1.



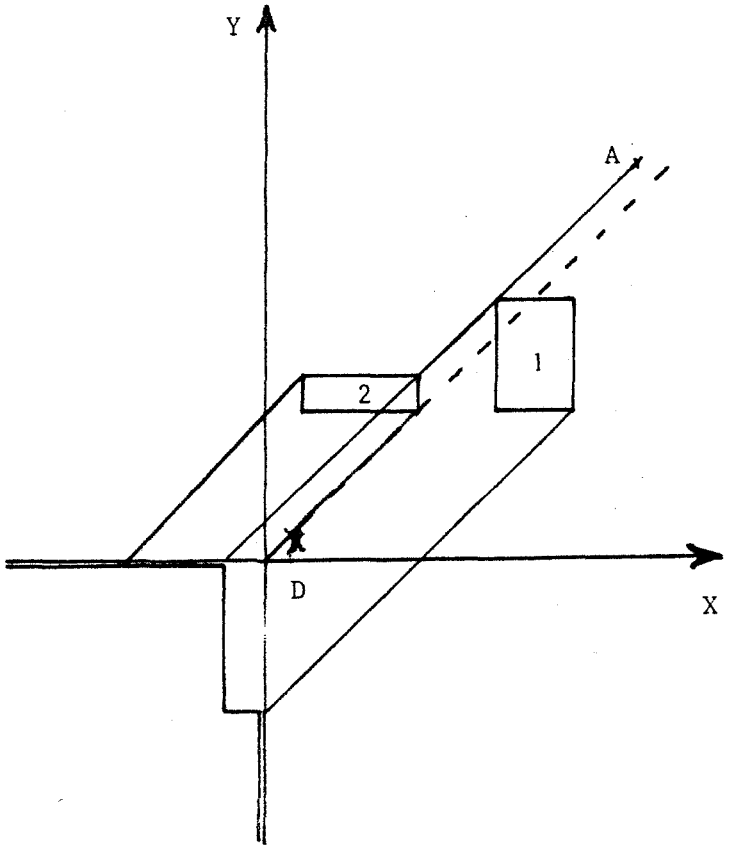
Retard du front d'onde après traitement des contraintes 1 puis 2.

Retard en A = 1

Par contre, si nous traitons d'abord 2, puis 1, nous obtiendrons les schémas suivants :



Retard du front d'onde après traitement de la contrainte 2.



Retard du front d'onde après traitement des contraintes 2 puis 1.

Retard en A = 0

Nous constatons que les retards que l'on peut lire sur les schémas de droite des pages précédentes sont différents selon l'ordre dans lequel nous avons présenté les contraintes à l'optimateur. Dans le problème posé, l'ordre d'introduction des contraintes sera celui que nous avons traité dans le second exemple, c'est-à-dire 2 puis 1. En réalité, cet ordre obéit à un algorithme naturel que nous pourrions énoncer de la façon suivante :

" Soient p contraintes C_1, C_2,.....C_p dans un plan, ces contraintes sont soumises à l'optimateur de telle sorte qu'une contrainte projetant une ombre sur une autre contrainte soit introduite dans l'optimateur avant cette dernière. L'ombre projetée par une contrainte étant définie comme la portion d'espace située derrière cette contrainte et comprise entre ses diagonales haute et basse. On définit ainsi une relation d'ordre sur les contraintes pour leur traitement par l'optimateur. Cette relation sera une permutation $C_{\alpha 1}$, $C_{\alpha 2}$, ..., $C_{\alpha p}$ des p contraintes du plan".

Remarquons que dans certains cas particuliers où l'exécution d'une tâche présente dans son déroulement un recouvrement des moyens générant plusieurs contraintes (cas de contraintes liées, figure 11), l'algorithme que nous venons de présenter est mis en défaut : si nous traitons les obstacles dans l'ordre préconisé par cet algorithme (contrainte 2 puis contrainte 1), le retard calculé pour le point A est inexact. Nous sommes alors amenés à "fusionner" les obstacles avant de les traiter (figure 12).

Ces cas particuliers sont assez rares et nous ne les avons jamais rencontrés dans les exemples pratiques que nous avons traités au chapitre 5.

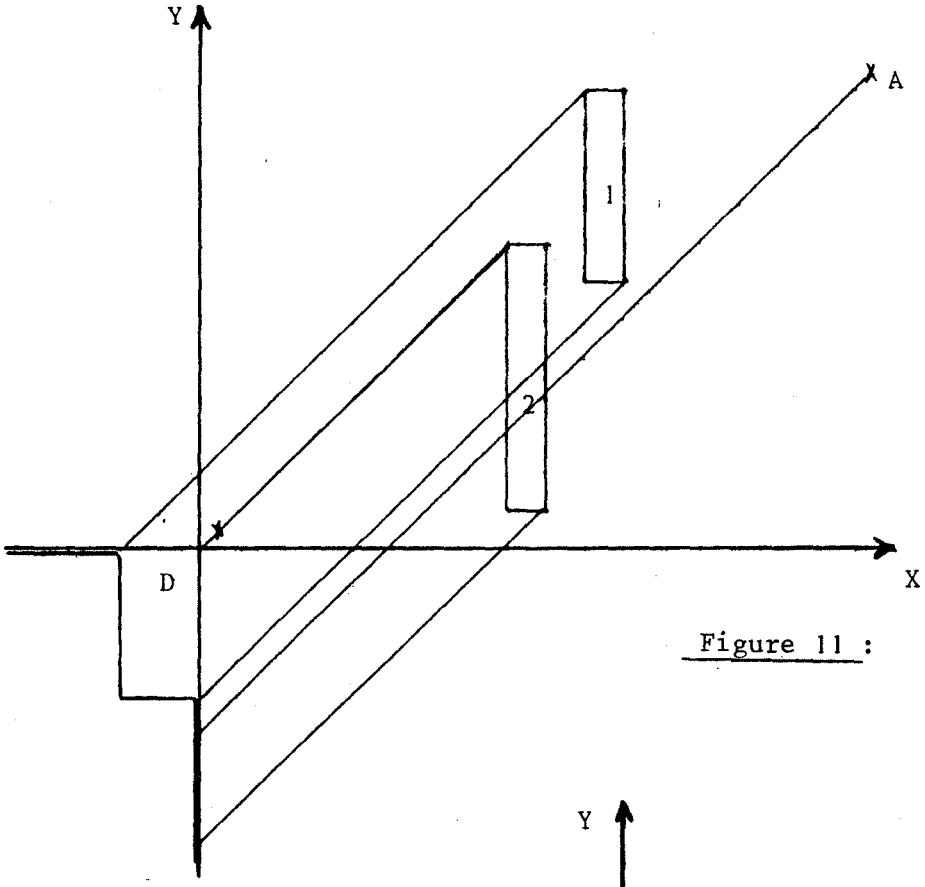


Figure 11 : Retard du front d'onde après traitement de la contrainte 2 puis 1.
Retard en A = 0

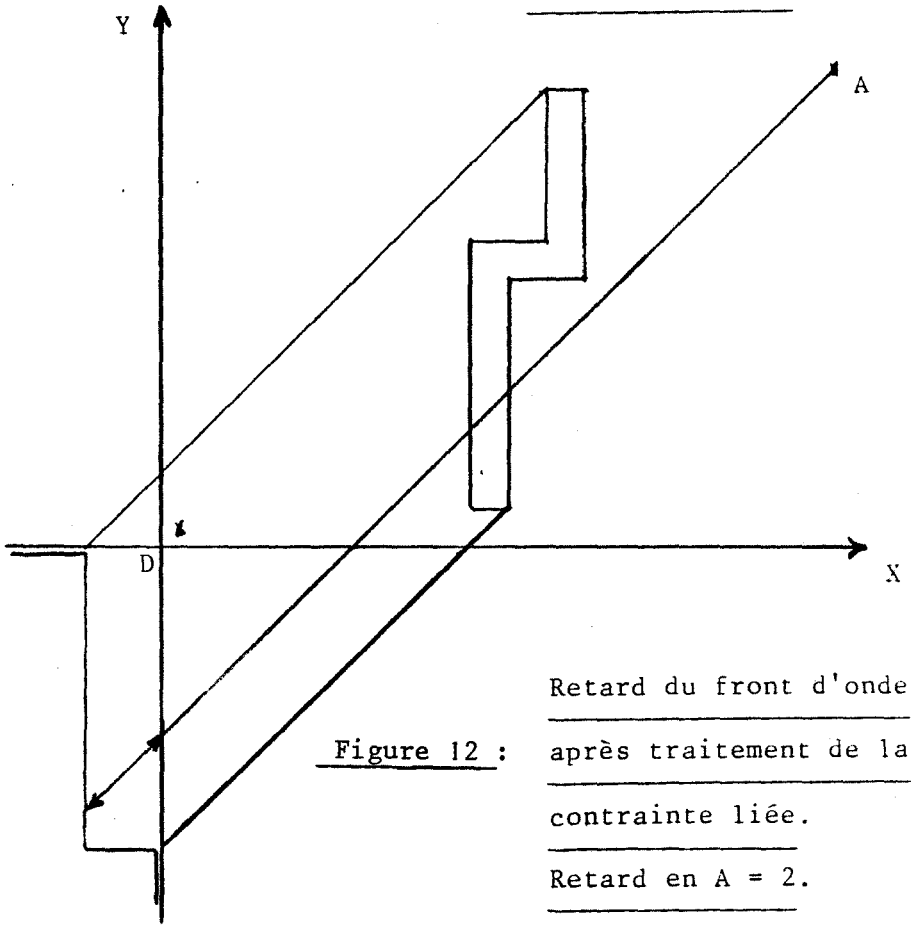


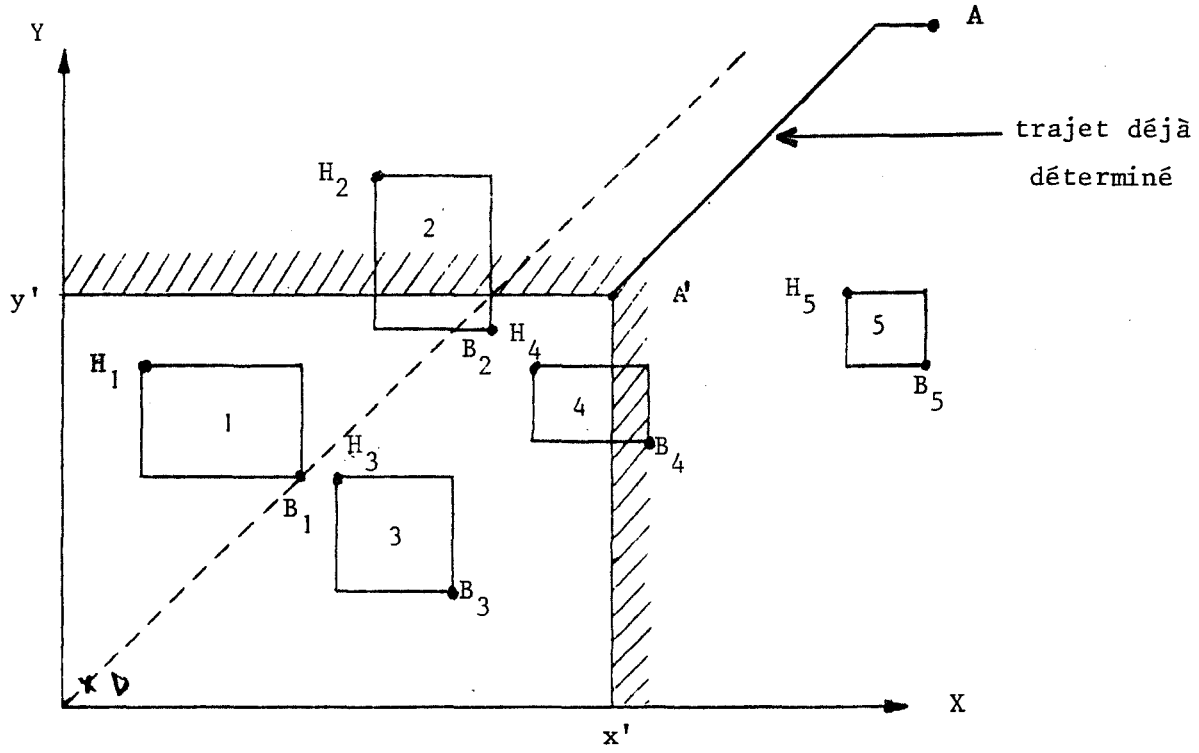
Figure 12 : Retard du front d'onde après traitement de la contrainte liée.
Retard en A = 2.

3-2-3- *Cas d'utilisation de l'optimateur* :

Comme nous le verrons lors de l'étude des recombinaisons associées à l'optimateur, la détermination du trajet spatial se fait à partir du point A vers le point D.

. Prise en compte des obstacles :

Lorsqu'au cours du problème, l'extrémité A' du trajet A'D restant à déterminer a une position quelconque par rapport aux obstacles, nous faisons subir à ceux-ci un traitement préalable.

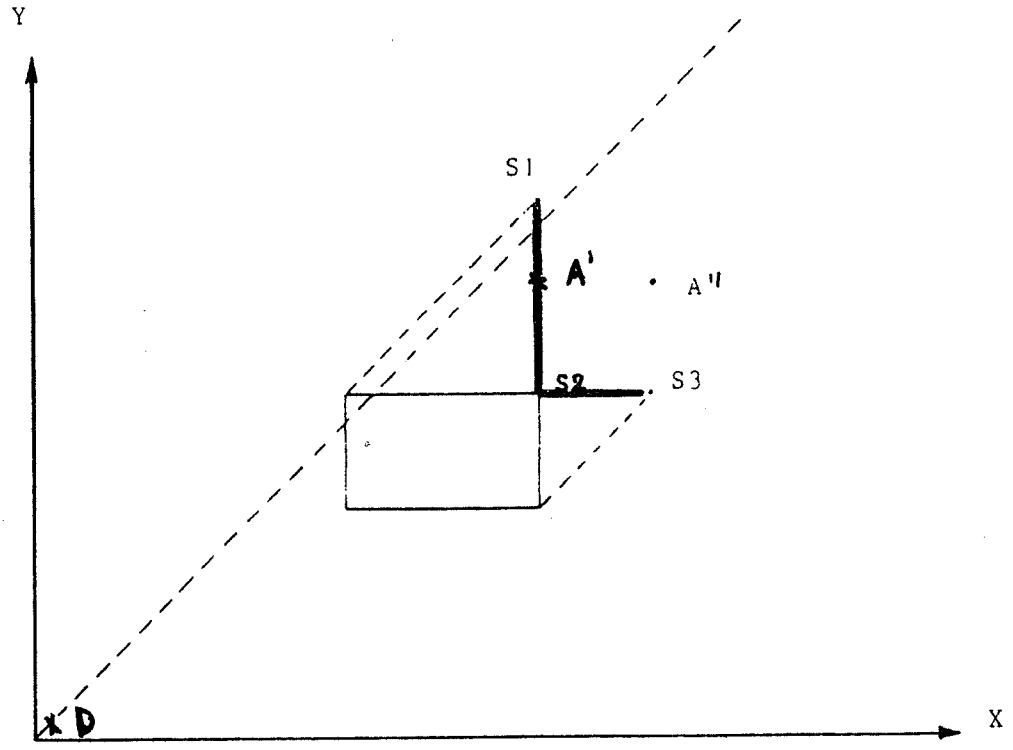


Comme en pratique, tout recul est interdit, la projection du trajet qui reste à définir, sera nécessairement à l'intérieur du rectangle non hachuré. Ce qui nous impose d'éliminer les contraintes complètement à l'extérieur de ce rectangle (ici N° 5), et de s'interdire de contourner les contraintes qui sont en partie à l'intérieur du rectangle (ici N° 2 et 4) en passant à l'extérieur de ce rectangle, pour cela nous allons minorer l'adresse basse de la contrainte N° 4 et majorer l'adresse haute de la contrainte N°2. Ainsi nécessairement le chemin passera entre les contraintes 2 et 4 .

Cette opération revient en fait à prolonger à l'infini vers les X positifs et les Y négatifs la contrainte 4, et à prolonger à l'infini vers les X négatifs et les Y positifs la contrainte 2, c'est-à-dire à créer des obstacles fictifs.

. Position du point A' nécessitant l'emploi de l'optimateur :

La décision importante à prendre pour chaque obstacle, est celle de son contournement (au-dessus ou au-dessous). Cette décision doit uniquement être prise si le point d'arrivée intermédiaire A' correspondant à l'état d'avancement du processus se trouve sur les segments S1-S2 ou S3-S2. Si le point est en A", intérieur à l'angle droit défini par S1-S2 et S2-S3, l'utilisation de l'optimateur est inutile car le choix du pas suivant ne détermine pas le trajet pour contourner l'obstacle. De même, si l'on se trouve à l'extérieur de cet angle droit, il est inutile de se servir de l'optimateur puisque dans ce cas, la décision de contourner l'obstacle par le dessus ou par le dessous est déjà prise.



. Formalisation :

Soient : - $H_i(x_{Hi}, y_{Hi})$ et $B_i(x_{Bi}, y_{Bi})$ les points haut et bas de la contrainte numéro i.

- $A'(x',y')$ le point extrémité du trajet restant à déterminer.

Pour chaque contrainte :

- Si : $x' \leq x_{Hi}$
  ou $y' \leq y_{Bi}$ élimination de la contrainte numéro i.

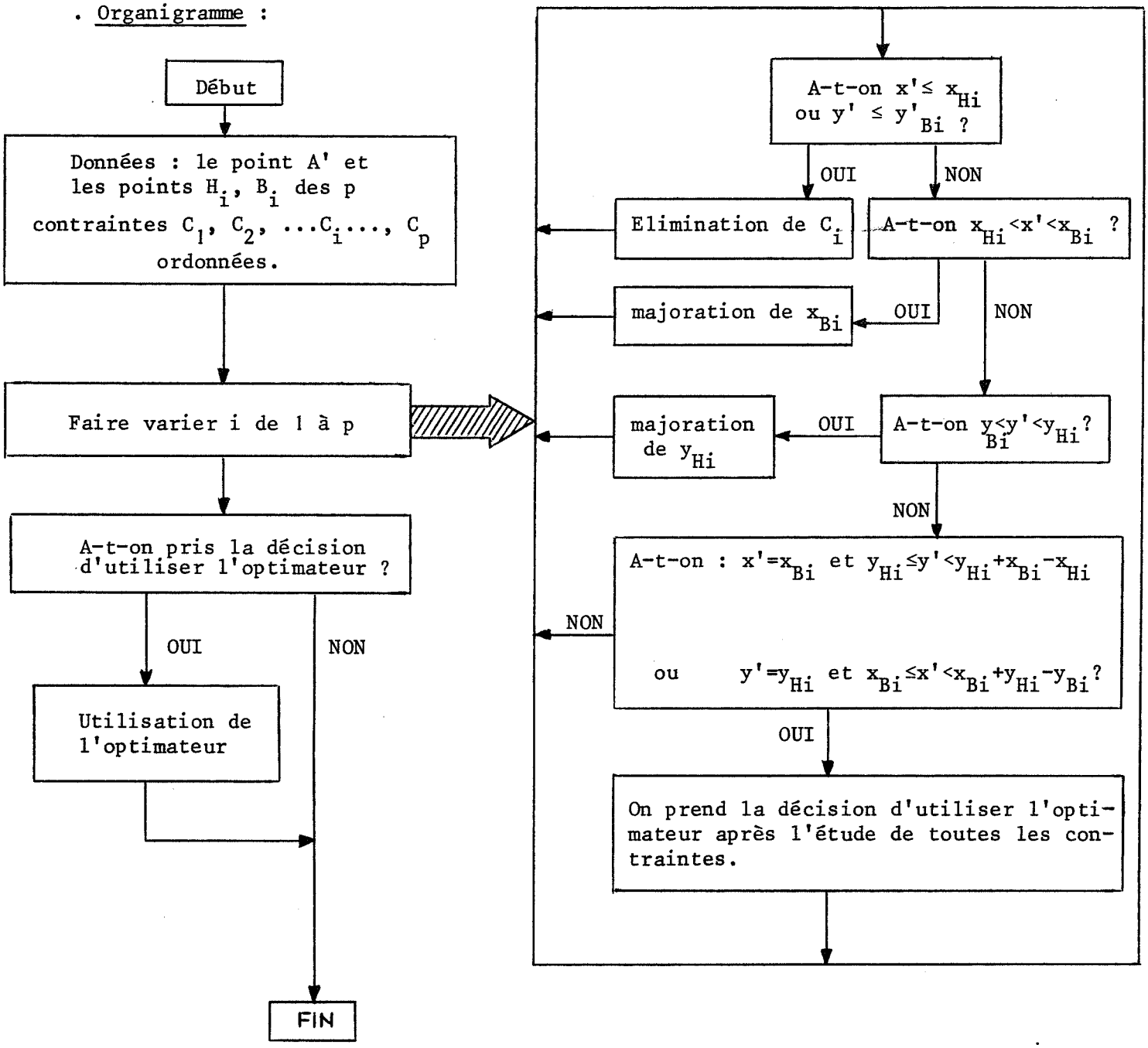
- Sinon :
  - Si $x_{Hi} < x' < x_{Bi}$ majoration de x_{Bi},
    ou $y_{Bi} < y' < y_{Hi}$ majoration de y_{Hi}.
  - Sinon conserver la contrainte telle quelle.

Ensuite, employer l'optimateur si pour au moins une des contraintes :

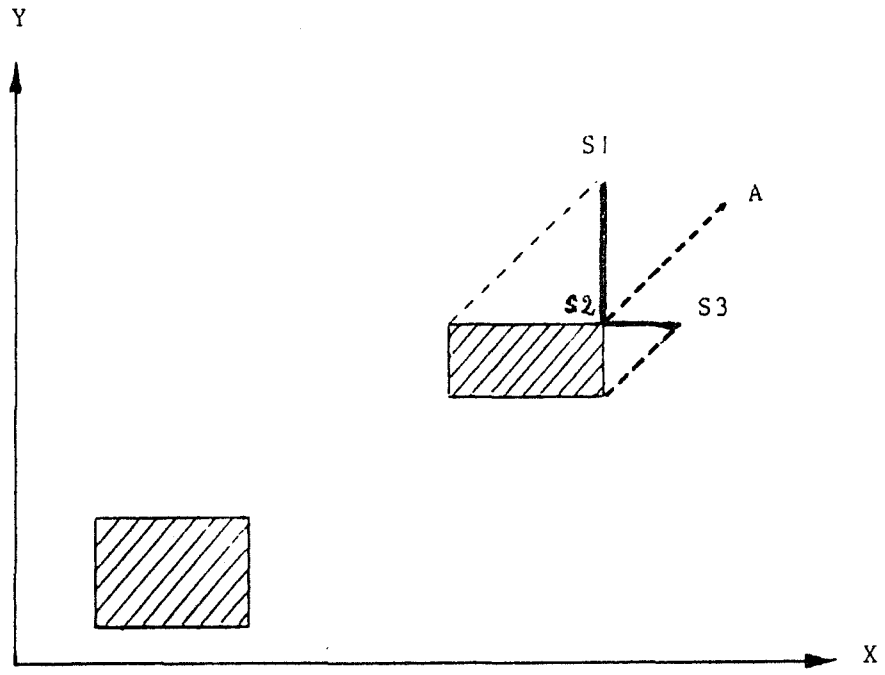
$$A' \in S_1S_2 \Longleftrightarrow x' = x_{Bi} \quad \text{et} \quad y_{Hi} \leq y' < y_{Hi} + x_{Bi} - x_{Hi}$$

$$\text{ou} \quad A' \in S_2S_3 \Longleftrightarrow y' = y_{Hi} \quad \text{et} \quad x_{Bi} \leq x' < x_{Bi} + y_{Hi} - y_{Bi}$$

. Organigramme :



3-2-4- *Portée d'une décision* :



Comme nous l'avons vu au paragraphe précédent, on ne doit prendre véritablement une décision que pour passer au-dessus ou au-dessous de chaque obstacle, donc lorsque l'on se trouve sur l'un des segments S1-S2 ou S3-S2 Si l'on ne se trouve pas sur l'un de ces segments relatifs à l'une quelconque des contraintes du plan considéré, on peut se déplacer sur une parallèle à la diagonale, c'est-à-dire faire progresser simultanément les deux processus X et Y, sauf si, bien entendu, dans d'autres plans de projection, certaines contraintes bloquent l'un ou l'autre de ces processus. Nous pouvons donc dire que la portée d'une décision va du franchissement des segments S1-S2 et S3-S2 d'une contrainte, jusqu'aux mêmes segments de la contrainte suivante figurant sur le trajet dans le plan considéré.

. Nombre d'activations de l'optimateur :

Compte tenu du fait qu'une décision n'est à prendre dans un plan donné que sur l'un des segments S1-S2 ou S2-S3 relatifs à un obstacle, le nombre total NA d'activations de l'optimateur est au plus égal au nombre total d'obstacles, étant donné qu'on prend au plus une décision par obstacle.

Comme le nombre moyen $\overline{NM}$ d'obstacles par plan est :

$$\overline{NM} = \frac{\mathcal{N}}{\frac{N(N-1)}{2}} = \frac{2\mathcal{N}}{N(N-1)}$$, le nombre total α d'introductions d'obstacles

dans l'optimateur à partir de l'ordinateur est :

$$\alpha = \mathcal{N} \times \frac{2\mathcal{N}}{N(N-1)} = \frac{2\mathcal{N}^2}{N(N-1)}$$

Exemple : $N = 40$, $\mathcal{N} = 2340 \Longrightarrow \alpha = 7020$

En conclusion, si le nombre d'obstacles est suffisamment faible pour qu'ils soient enregistrés entièrement en mémoire centrale, le temps passé à transmettre les obstacles de l'ordinateur à l'optimateur est de l'ordre de la seconde, et négligeable par rapport à la durée des autres calculs de l'ordinateur.

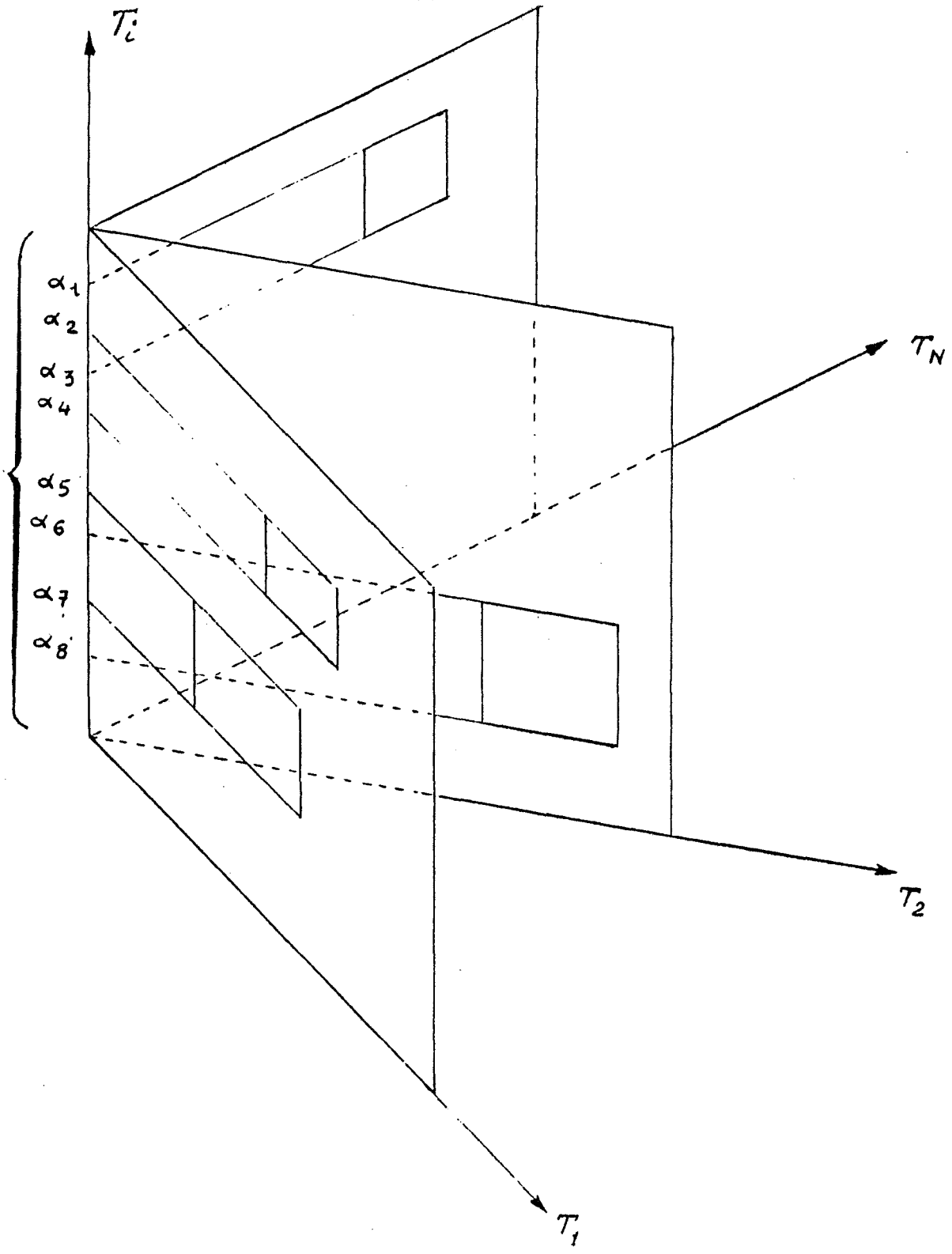
. Algorithme de recherche du point où se situe la prochaine prise de décision :

Si on désire connaître le nombre de pas que l'on peut construire dans l'espace après avoir utilisé l'optimateur et avant de rencontrer un nouvel obstacle, il est possible de comparer la position du dernier point construit de la trajectoire à tous les $\mathcal{N}$ obstacles. Comme cette opération devrait être répétée au maximum $\mathcal{N}$ fois, il faudrait réaliser $\mathcal{N}^2$ comparaisons entre cette position et celle d'un obstacle, ce qui est énorme.

Exemple : $\mathcal{N} = 2340 \Longrightarrow \mathcal{N}^2 = 4\ 800\ 000.$

Il nous parait préférable d'utiliser la méthode suivante : pour chacun des N axes de coordonnées T_i (i = 1,2,...,N),construisons une table dans laquelle nous portons la valeur de la i-ème coordonnée du début et de la fin de chaque obstacle situé dans les N-1 plans $T_iT_1, T_iT_2, \ldots, T_iT_N$. Trions ensuite ces valeurs par ordre décroissant, nous obtenons ainsi une liste des "événements" rencontrés par la i-ème coordonnée du point d'arrivée qui se déplace de l'extrémité vers l'origine au cours de la recombinaison. De la sorte, pour toute position du mobile, nous pourrons déterminer rapidement le prochain évènement réclamant une action au cours de la résolution du problème.

Comme les nombres à trier sont tous entiers et compris entre 0 et 1023 (l'optimateur testé prenant en compte les coordonnées entières variant entre 0 et 1023 uniquement), la méthode de tri par clé semble être bien adaptée à ce genre de problème.



évènements relatifs à la i-ème coordonnée

Soit $\overline{NT}$ le nombre moyen d'obstacles projetés sur un axe T_i ;comme le nombre moyen d'obstacles par plan est $\overline{NM} = \frac{2\mathcal{L}}{N(N-1)}$ et comme chaque T_i intervient dans (N-1) plans, il y a en moyenne :

$$\overline{NT} = (N-1)\,\frac{2\mathcal{L}}{N(N-1)} = \frac{2\mathcal{L}}{N}$$

obstacles projetés sur l'axe T_i, et comme chaque obstacle a 2 points particuliers (début et fin) ceci représente 2 $\overline{NT}$ points en moyenne par axe, soit $\frac{4\mathcal{L}}{N}$.

Prenons un exemple numérique : N = 40, $\mathfrak{N}$ = 2340 (soit en moyenne 3 obstacles par plan). Alors $\frac{4\mathfrak{N}}{N}$ = 234, il y a donc 40 fois 234 valeurs à trier, chaque valeur étant comprise entre 0 et 1023. Une méthode de tri par fusion nécessiterait :

$$40 \times 234 \times \log_2 234 \simeq 80\ 000 \text{ opérations.}$$

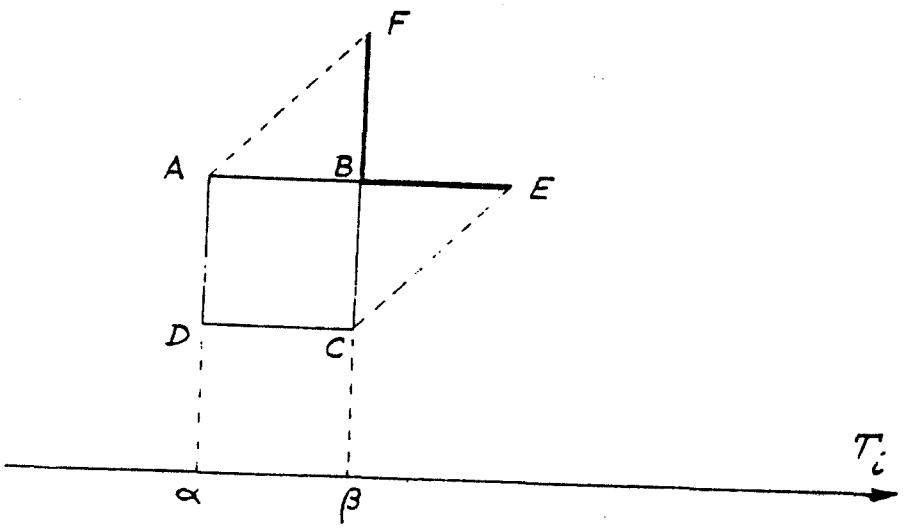
Une méthode de tri par clé en triant d'abord sur les 5 derniers bits, puis sur les 5 premiers, nécessiterait :

$$40 \times 2 \times 234 \simeq 20\ 000 \text{ opérations.}$$

Cette dernière méthode semble donc la plus rapide dans notre exemple, mais elle est malheureusement coûteuse en place (il est nécessaire de chaîner les éléments en listes). Si la place mémoire manquait, le recours à la méthode de tri par fusion pourrait s'avérer nécessaire.

Actions requises à la rencontre d'un évènement :

Ces actions peuvent être diverses.



Supposons que la coordonnée relative au processus i passe par la valeur β au cours de la construction de la trajectoire. Cette trajectoire coupe par conséquent l'hyperplan de coordonnée β . L'action requise peut alors prendre différentes formes.

Si la trajectoire passe au-dessous de C ou au-dessus de F, aucune action n'est requise.

Si la trajectoire passe entre B et C, il y a lieu de bloquer la coordonnée i du processus.

Si la trajectoire passe entre B et F, il faut activer l'optimateur pour déterminer la longueur du trajet minimum dans le plan considéré.

Au moment où la i-ème coordonnée passe par la valeur α, il y a lieu d'effacer complètement l'obstacle ABCD de la liste des obstacles à prendre en compte.

3-3. *INTRODUCTION DES DONNEES DANS L'OPTIMATEUR* :

L'introduction des données peut être réalisée soit manuellement par des clés, soit par une interface. Les résultats sont affichés par des voyants et sont rendus disponibles sur l'interface. L'optimateur testé à l'Ecole des Mines de Saint-Etienne comporte 2048 cellules.

3-3-1- *Fonctionnement manuel* :

Le panneau avant est pourvu d'un dispositif de visualisation de la valeur binaire des adresses basse ou haute, de la valeur binaire du retard occasionné par les contraintes et du signal fin de calcul. Un ensemble d'interrupteurs permet de tester l'appareil par introduction manuelle des données (adresse basse puis adresse haute).

3-3-2- *Fonctionnement en connection à un ordinateur* :

L'utilisation exclusive des circuits intégrés de technologie TTL confère à l'appareil une compatibilité totale avec les ordinateurs disponibles sur le marché français. Il peut être relié à l'un de ces appareils par un cordon qui permet d'en recevoir des mots de 16 bits. Ces bits sont répartis de la façon suivante :

- le premier qui est désigné par "servi" joue le rôle de signal d'horloge pour le registre-tampon,
- les quatre suivants correspondent respectivement à l'adresse basse, à l'adresse haute, à l'initialisation et à la sortie du résultat,
- les onze derniers définissent une adresse.

1 - Cycle d'initialisation :

Avant de commencer le calcul il convient d'initialiser l'optimateur, c'est-à-dire d'écrire des "0" depuis l'adresse 0 jusqu'à l'adresse 1023 puis des "1" de l'adresse 1024 à l'adresse 2047. Pour cela on met l'adresse basse à 0 si bien que le mot envoyé par l'ordinateur est le suivant :

Servi	B	H	I	Re	2^{10}	2^9	2^0
1	1	0	0	0	0	0	0

On met ensuite l'adresse haute à 2047, l'ordinateur envoie donc :

Servi	B	H	I	Re	2^{10}	2^9	2^0
1	0	1	1	0	1	1	1

C'est la valeur du bit I qui enclenche le séquenceur sur la mise à 0 depuis l'adresse basse c'est-à-dire 0 jusqu'à 1023, puis à 1 depuis 1024 jusqu'à l'adresse 2047. Ensuite l'optimateur émet un signal "Fin de Cycle", ce qui indique qu'il est prêt à recevoir les données relatives au traitement d'une première contrainte.

2 - Cycle normal :

Le signal "fin de cycle" dont il vient d'être question interrompt l'ordinateur qui enregistre le signal. S'il n'est pas prêt à envoyer les deux adresses qui définissent la première contrainte, il poursuit ses calculs et l'optimateur reste en attente. Dans le cas contraire il envoie successivement deux mots de 16 bits qui correspondent aux adresses de cette première contrainte. Il envoie donc :

- 1er mot

Servi	B	H	I	R	2^{10} ..... 2^0
1	1	0	0	0	adresse basse

- 2ème mot

Servi	B	H	I	Re	2^{10} ..... 2^0
1	0	1	0	0	adresse haute

Dès que l'adresse haute est chargée, le séquenceur déclenche un cycle de calcul.

Pendant ce travail l'ordinateur entreprend les calculs relatifs à la contrainte suivante.

Lorsque l'optimateur a terminé le calcul du retard introduit par la première contrainte, il émet de nouveau le signal "Fin de cycle", ce qui a pour effet d'interrompre l'ordinateur qui, là encore, dès qu'il est prêt, transmet à l'optimateur les deux adresses qui définissent la seconde contrainte, etc...

3 - Cycle résultat :

Lorsque l'ordinateur reçoit le signal "Fin de cycle" de ce qu'il sait correspondre à la dernière contrainte, il demande à l'optimateur d'effectuer un cycle résultat. Nous rappelons qu'il s'agit d'effectuer la somme des "1" de l'adresse 0 jusqu'à l'adresse d'arrivée A incluse.

A cet effet l'ordinateur charge le registre d'adresse basse à 0 et celui d'adresse haute à A. Il envoie donc les deux mots suivants :

- 1er mot

Servi	B	H	I	Re	2^{10}	2^0
1	1	0	0	0	0	0

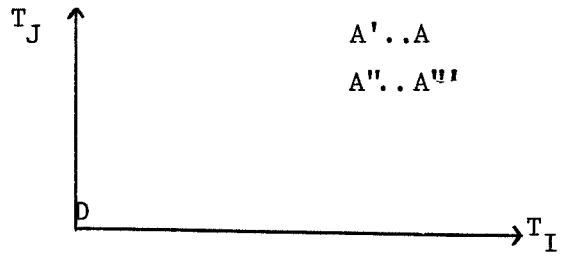
- 2ème mot

Servi	B	H	I	Re	2^{10}	2^0
1	0	1	0	1	Valeur de l'adresse arrivée	

La valeur 1 du bit Re fait effectuer la somme voulue et, lorsque cette somme se trouve dans le registre de l'additionneur, l'optimateur envoie le signal "Fin de cycle". La carte de couplage est connectée directement au registre de l'additionneur si bien que le résultat s'y trouve disponible tant qu'un nouveau calcul n'est pas venu modifier le contenu de ce registre.

3 - 4. *RECOMBINAISONS ASSOCIEES A L'OPTIMATEUR* :

Les données sur lesquelles travaillent ces recombinaisons sont, comme nous l'avons indiqué, les nombres de cycles d'horloge nécessaires pour atteindre les trois prédécesseurs A', A", A"' du point d'arrivée dans chaque plan (T_I, T_J).



3-4-1- *Recombinaison majoritaire* :

Cet algorithme est très simple et malgré cela, il conduit à des résultats comparables à ceux obtenus par d'autres recombinaisons, comme nous le verrons dans le dernier chapitre.

. Principe :

Son principe consiste à examiner dans chacun des plans quel est le point (A', A" ou A"') qui est atteint depuis D en un nombre minimal de cycles d'horloge. En reprenant les notations du paragraphe 3-1-3, nous avons :

- $T_p(A')$, temps (ou cycles d'horloge) nécessaire pour atteindre A',
- $T_p(A")$, temps (ou cycles d'horloge) nécessaire pour atteindre A",
- $T_p(A"')$, temps (ou cycles d'horloge) nécessaire pour atteindre A"',

A chaque processus T_I, ($I\epsilon\{1,....,N\}$), est associé un compteur C(I) et une variable d'incompatibilité INC(I), tous deux initialisés à 0 ; on examine chacun des plans de projection en effectuant les opérations suivantes :

- si $T_p(A') = INF\,[T_p(A'), T_p(A"), T_p(A"')]$, la trajectoire la plus courte passe par AA' ; on a donc intérêt à faire progresser le processus T_I seul. C(I) est incrémenté du temps relatif à A"' diminué du temps relatif à A' : $C(I) \leftarrow C(I)+T_p(A"')-T_p(A')$,

- si $T_p(A''') = INF\,[T_p(A'), T_p(A''), T_p(A''')]$, on incrémente C(J) du temps relatif à A' diminué du temps relatif à A''' :

$$C(J) \leftarrow C(J) + T_p(A') - T_p(A'''),$$

- si $T_p(A'') = INF\,[T_p(A'), T_p(A''), T_p(A''')]$, ce qui incite à faire avancer simultanément T_I et T_J, C(I) et C(J) restent inchangés.

- si le point A' est interdit (pénétration dans une contrainte), la variable d'incompatibilité relative à T_I est mise à 1 :

$$INC\ (I) \leftarrow 1$$

- si le point A''' est interdit : INC (J) ← 1.

Lorsque tous les plans ont été examinés, on classe les valeurs C(1), C(2),...,C(N) par ordre décroissant et on choisit le processus T_K pour lequel C(K) est maximal et INC(K) $\neq$ 1. On fait progresser T_K d'un pas : si x_K est la coordonnée du point A, sa nouvelle coordonnée sera x_K-1. Il est alors nécessaire de passer en revue les N-1 plans où figure ce processus pour tenir compte du cas particulier suivant : supposons que dans l'un des plans (T_K, T_L), le point A'' soit interdit (noté $T_p(A'') = \infty$) sans que A' et A''' le soient (cas d'une contrainte ayant un point anguleux en A). La progression de l'un des processus isolément est permise, mais leur progression simultanée est interdite. Dans la nouvelle situation déterminée par l'avance de T_K, la progression de T_L n'est plus possible, ce qui doit se traduire par l'affectation de la valeur 1 à la variable INC(L) :

$$T_p(A'') = \infty \Longrightarrow INC(K) = 1 \text{ ou } INC(L) = 1.$$

On cherche ensuite le processus $T_{K'}$ tel que C(K') soit immédiatement inférieur à C(K) et tel que INC(K') $\neq$ 1. On fait progresser $T_{K'}$ (la coordonnée $x_{K'}$ devient $x_{K'}$-1) et on examine les plans où il figure pour voir s'il n'y a pas de cas nécessitant la mise à 1 de certaines variables d'incompatibilité. On itère jusqu'à épuisement des processus.

Le pas est complètement déterminé lorsque tous les processus ont progressé, ou lorsque les variables d'incompatibilité associées à ceux qui n'ont pas progressé sont toutes égales à 1.

Pour passer à l'étude du pas suivant, il est nécessaire de procéder à la remise à 0 des tableaux C(I) et INC(I) et à un nouvel examen des plans de projection. Le problème est résolu lorsque le point figuratif de l'état des processus coïncide avec D.

Structure des données :

L'algorithme précédemment décrit programmé en PL1600 occupe environ 1 K mots de 16 bits sur l'ordinateur T1600. Sa mise en oeuvre nécessite en outre la réservation de mémoire pour y stocker les données et les tableaux de travail. Dans le cas où le nombre de processus est N et le nombre de contraintes par plan K, ces tableaux ont les tailles suivantes :

- stockage des données :

. N mots pour définir les coordonnées du point d'arrivée dans l'espace.
. $4\,K\,\frac{N(N-1)}{2}$ mots pour définir les coordonnées cartésiennes des contraintes (4 mots par contrainte).
. $4\,\frac{N(N-1)}{2}$ mots qui définissent dans chaque plan son numéro, le nombre des contraintes, et les deux processus concernés.

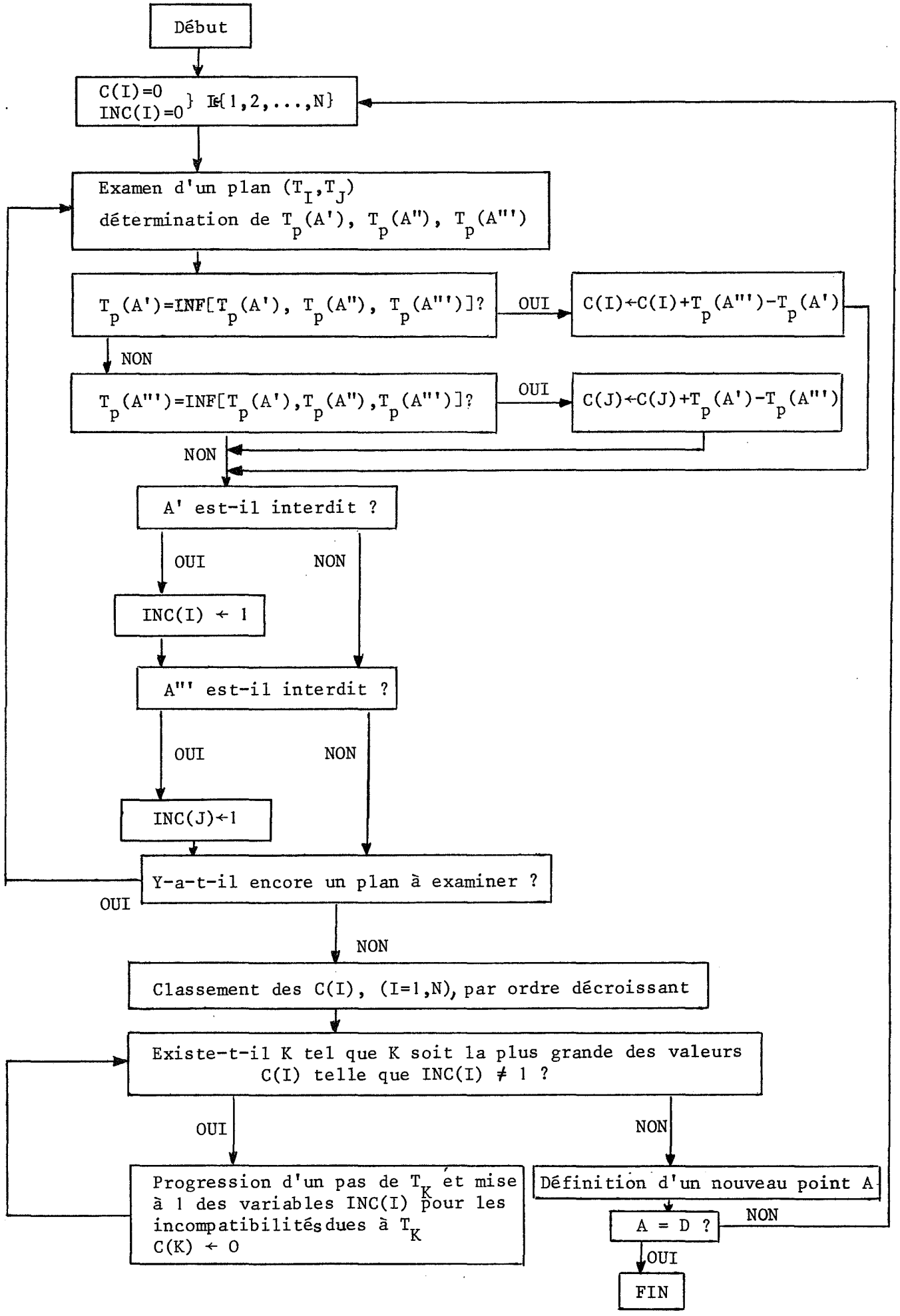
- tableaux de travail :

. N mots qui correspondent aux compteurs C(I) .
. N mots qui correspondent aux variables d'incompatibilité INC(I).
. N mots pour trier les processus à la suite de l'analyse bidimensionnelle.
. $\frac{N(N-1)}{2}$ mots pour indiquer si la progression simultanée des deux processus associés à chaque plan est permise.

Soit en tout $2N(N-1)\cdot(K+\frac{5}{4}) + 4N \neq 2N^2(K+1)$ mots.

La résolution d'un problème à 20 processus et 5 obstacles par plan exige que la mémoire disponible dans l'ordinateur soit au moins de 6 K mots.

. Organigramme :



3-4-2- Recombinaison arborescente :

Nous faisons l'hypothèse que la longueur de la trajectoire à N dimensions n'est que légèrement supérieure à celle de la plus longue trajectoire plane. La méthode employée consiste donc à minimiser la plus longue trajectoire plane.

Pour un problème de N processus à ordonnancer, nous aurons 2^N-1 points possibles pour placer l'extrémité du pas à définir.

Ces points étant ceux qui précèdent immédiatement le point d'arrivée dans l'espace à N dimensions.

Soit M l'un quelconque de ces 2^N-1 points. L'optimateur ou toute autre machine bidimensionnelle permet de déterminer la distance dans un plan entre le point de départ et la projection de M dans ce plan.

Pour chaque plan la machine fournit 4 valeurs correspondant aux 4 projections possibles A, A', A", A"' de M. Au total, on obtient donc $4 \times \frac{N(N-1)}{2} = 2N(N-1)$ valeurs qui sont les données utilisées dans la recombinaison.

Appelons d la distance DM que l'on cherche à minimiser.

Posons $K = \frac{N(N-1)}{2}$. Considérons le plan n° i ($1 \leq i \leq K$).

$T_p^i(A)$, $T_p^i(A')$, $T_p^i(A")$, $T_p^i(A"')$ sont les longueurs des trajectoires planes DA, DA', DA", DA"'.

Nous savons que d sera supérieure ou égale à la plus grande des valeurs de l'ensemble E tel que :

$$E = \{T_p^1(A),\ T_p^1(A'),\ T_p^1(A"),\ T_p^1(A"'),$$
$$T_p^2(A),\ T_p^2(A'),\ T_p^2(A"),\ T_p^2(A"'),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$T_p^K(A),\ T_p^K(A'),\ T_p^K(A"),\ T_p^K(A"')\}$$

Le problème consistera donc à supprimer parmi ces 4 K valeurs, toutes les plus grandes, en respectant bien sûr la compatibilité qui doit exister entre elles (pour qu'il existe toujours au moins une solution physiquement réalisable).

Nous obtiendrons finalement un ensemble de K valeurs, à savoir une par plan et nous pourrons seulement dire que d sera supérieure à cette distance, en espérant qu'elle ne sera que légèrement supérieure. Pour rendre compte des impossibilités (pénétration dans une contrainte), nous introduirons des distances infinies.

. Algorithme :

1 - Déterminer les 2N(N-1) éléments de l'ensemble E.
2 - Classer les éléments de E par ordre décroissant.
3 - Considérer l'arbre binaire formé par toutes les solutions (tous les éléments de E).
4 - Supprimer progressivement toutes les branches correspondant aux plus grands éléments de E, s'arrêter lorsqu'il ne reste plus qu'une seule solution.
5 - Cette solution définit un point M parmi ces 2^N-1 points prédécesseurs du point d'arrivée. Ce point M devient le nouveau point d'arrivée.
6 - Si M ≠ D, retour en 1 pour définir un autre élément de la trajectoire, sinon la trajectoire est déterminée.

. Exemple :

Pour fixer les idées, nous allons traiter un exemple simple avec 3 processus.

Considérons trois processus notés 1,2,3. L'optimateur CYBCO nous donne pour chacun des couples de processus les distances correspondant aux 4 états possibles : les deux processus progressent (noté 1-2), l'un des deux seulement progresse (noté $1-\bar{2}$ ou $\bar{1}-2$) ou aucun (noté $\bar{1}-\bar{2}$).

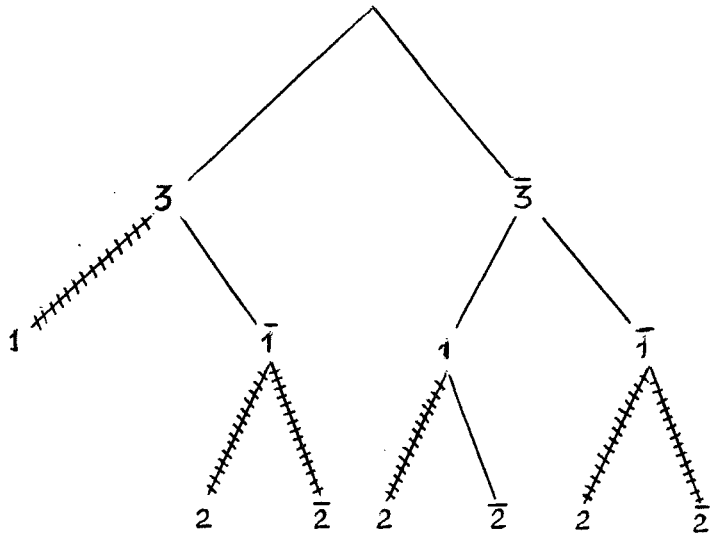
Processus considérés	distances	
1-3	∞	d_2
$1-\bar{3}$	10	
$\bar{1}-3$	13	
$\bar{1}-\bar{3}$	11	
2-3	5	d_1
$2-\bar{3}$	15	
$\bar{2}-3$	11	
$\bar{2}-\bar{3}$	12	
1-2	∞	d_3
$1-\bar{2}$	12	
$\bar{1}-2$	11	
$\bar{1}-\bar{2}$	12	

Nous choisissons parmi toutes ces valeurs la plus grande (ici, 1-3), et nous commençons à construire un arbre avec les branches 1-3, $\bar{1}-3$, $\bar{3}-1$ et $\bar{3}-\bar{1}$ parmi lesquelles nous effaçons déjà la branche 1-3 parce que sa valeur est la plus grande.

Prenons maintenant la valeur suivante ; ce sera par ordre décroissant 1-2. Ici, il suffira d'ajouter un niveau dans l'arbre correspondant au processus 2 puisque le niveau correspondant au processus 1 figure déjà dans l'arbre. Supprimons maintenant les éventualités correspondant à l'état 1-2.

Pour la troisième distance, ici $2-\bar{3}$, nous n'ajouterons pas de niveau dans l'arbre puisque les 2 processus figurent déjà dans l'arbre ; nous supprimons simplement les branches correspondant à $2-\bar{3}$. Continuons l'étude avec $\bar{1}-3$. Il ne nous reste plus que la solution $\bar{3}-1-\bar{2}$ puisque la solution $\bar{3}-\bar{1}-\bar{2}$, correspondant à l'arrêt de tous les processus est rejetée d'office.

En réalité, le nombre de processus peut être très grand. Très vite, l'arbre deviendrait trop volumineux pour résider en mémoire centrale de l'ordinateur et la recherche en arbre prendrait trop de temps.



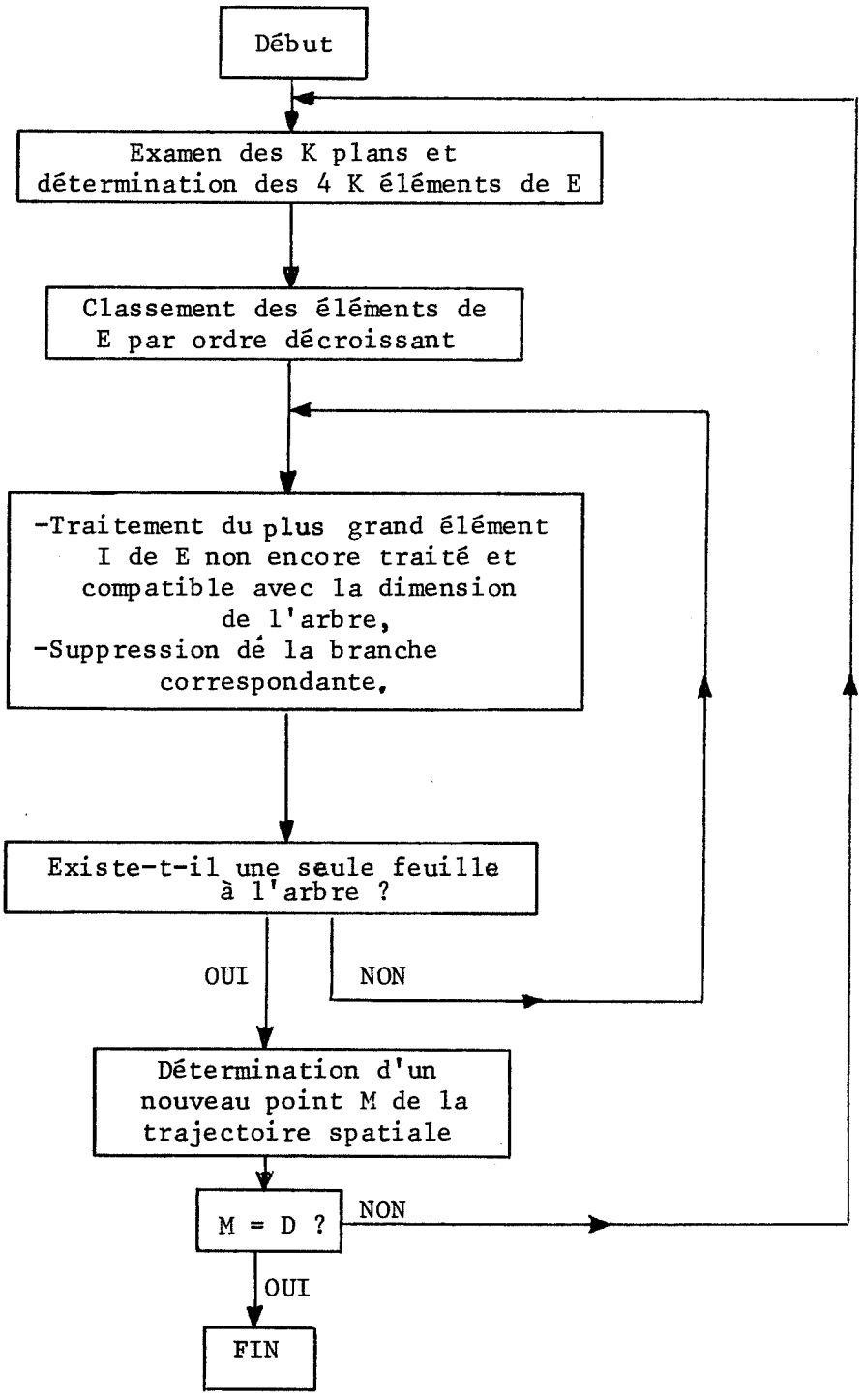
Nous constatons que le fait de sélectionner une distance relative à deux nouveaux processus multiplie par 4 le nombre de feuilles de l'arbre avant qu'on puisse en supprimer une ou plusieurs.

Si par contre la distance choisie ne fait intervenir qu'un seul nouveau processus, le nombre de feuilles est simplement multiplié par 2. Enfin, le fait de prendre en compte une distance telle que les processus concernés figurent déjà dans l'arbre permet d'en réduire immédiatement la taille.

Par conséquent, lorsque le nombre de branches devient trop grand et que la distance que l'on devrait considérer correspond à un ou deux processus, absents de l'arbre, on recherche la plus grande distance immédiatement inférieure mais permettant de faire diminuer le nombre de branches. Lorsque ce nombre est redevenu compatible avec la taille de la mémoire centrale de l'ordinateur et avec le temps que l'on peut consacrer à la recombinaison, on revient aux distances plus grandes que l'on avait laissées en attente.

Cette méthode permet donc d'obtenir une solution par élimination de toutes les distances les plus longues, en respectant la compatibilité. Elle est aussi très souple car en modifiant le nombre maximal de feuilles que l'on peut tolérer, on agit sur la taille de la mémoire centrale que l'on utilise et aussi sur le temps d'exécution de la recombinaison.

. Organigramme :



C'est finalement la première de ces deux recombinaisons (la recombinaison majoritaire) qui a été implémentée sur l'ordinateur Télémécanique T1600 pour tester l'optimateur.

4 - ETUDE DU COMBINATEUR OPTIMISANT SNCF-GENETE [12]

Il s'agit d'une machine capable de fournir, dans des temps très brefs, une solution à des problèmes de type combinatoire dans lesquels peuvent intervenir de nombreuses variables pouvant prendre de multiples états.

Cette machine appelée en réalité par ses inventeurs "combinateur hybride optimisant" est un combinateur car elle combine de nombreuses variables entre elles en vue d'obtenir une solution réalisable d'un problème d'ordonnancement.

Ce combinateur est dit optimisant car il a pour fonction de donner une solution aussi voisine que possible de la solution optimale.

Cependant, dans l'étude que nous en avons faite, nous n'emploierons jamais le mot "hybride". En effet, cette terminologie est assez mal choisie : le combinateur utilise des techniques numériques seules et non à la fois numériques et analogiques, il n'y a donc pas lieu de parler de combinateur hybride. Cette appellation a sans doute été un choix historique, venant des recherches du Docteur SAUVAN qui étudiait l'analogie avec les circuits neuronaux.

Le combinateur optimisant, décrit dans le brevet SNCF-M.GENETE, comprend :

A - Le système d'entrée et l'interface d'entrée :

Ces entrées peuvent être . des entrées directes,
. des entrées en temps réel,
. des entrées en temps partagé.

B - Les 6 opérateurs fonctionnels, comprenant :

- l'opérateur de pilotage OP.

Cet opérateur est câblé et programmé, il intègre le combinateur dans la chaîne de machines résolvant le problème et gère le système de sortie.

- l'opérateur directeur OD.

Cet opérateur est câblé et programmé, il met en relation les paramètres du problème et les variables du combinateur, synchronise et enchaîne logiquement les différentes opérations de chacun des autres opérateurs fonctionnels.

- l'opérateur analyse OA.

Il est câblé et détermine le coût des chemins optimaux dans les plans d'analyse.

- l'opérateur recombinaison OR.

Il est câblé et programmé, traite les coûts des chemins optimaux dans les plans d'analyse. Il construit une trajectoire spatiale unique, voisine de l'optimale, en évitant les hypervolumes de contraintes.

- l'opérateur compatibilité OC.

Il est câblé et élimine les solutions dont les composantes feraient pénétrer la trajectoire dans des hypervolumes de l'espace.

- l'opérateur spécialisé OS.

Il est câblé et programmé, intègre différentes variantes de structure du combinateur, au choix de l'opérateur et selon la nature du problème posé.

C - Le système de sortie et l'interface de sortie :

Ces sorties peuvent être :

- des sorties directes en temps réel,
- des sorties classiques.

Les phénomènes dynamiques sont représentés par une suite discrète d'états, appelés situations S(1), S(2),....S(i)..., appartenant à un espace pluridimensionnel. S(i) est l'extrémité d'un vecteur ayant pour composantes les états [S(i)X, S(i)Y, S(i)Z, S(i)T,...] des différentes variables mises en jeu. Une variable ne peut évoluer que selon une dimension. L'enchaînement des différentes situations S(1), S(2),... S(i)... constitue la "trajectoire" ou la stratégie.

Cette trajectoire est définie par une ligne brisée formée par une chaîne de vecteurs de modules l'unité, construite dans un espace n-dimensionnel si le nombre de variables est n.

Au cours d'une action élémentaire, la variable X peut :

- effectuer une action 1 en passant de l'état S(i)X à l'état S(i+1)X.

- effectuer une action 0, en restant dans l'état S(i)X = S(i+1)X.

L'action complexe ou "pas" est la combinaison de n actions élémentaires si le problème est à n variables, une action élémentaire par variable.

Nous appellerons :

. SD le point représentatif de la situation origine du problème,

. SA le point représentatif de la situation arrivée ou fin du problème.

Lorsque certaines actions élémentaires sont incompatibles entre elles, on dira qu'il existe une contrainte entre ces actions.

Si la contrainte entre deux actions de deux variables n'est pas modifiée par l'évolution d'une variable sur un autre axe (cas le plus général), elle est représentée dans l'espace n-dimensionnel par un cylindre de hauteur illimitée, orthogonal au plan formé par les deux axes représentant les deux variables dont certains couples d'actions sont interdits.

Si, au contraire, la contrainte entre deux actions de deux variables est modifiée par l'évolution d'une variable sur un autre axe, elle est représentée par un hypervolume isolé dans l'espace n-dimensionnel.

Le combinateur permet de déterminer un chemin de cet espace n-dimensionnel qui relie au mieux les points SD et SA en contournant les hypervolumes de contraintes. Pour cela la machine détermine des chemins optimaux dans chacun des C_n^2 plans de projection du problème, ces chemins seront ensuite recombinés entre eux pour obtenir la trajectoire spatiale.

Nous allons maintenant décrire les principaux organes du combinateur, ce qui permettra de voir comment cette machine peut résoudre un problème d'ordonnancement.

4-1- L'opérateur analyse :

La résolution d'un problème par le combinateur nécessite :

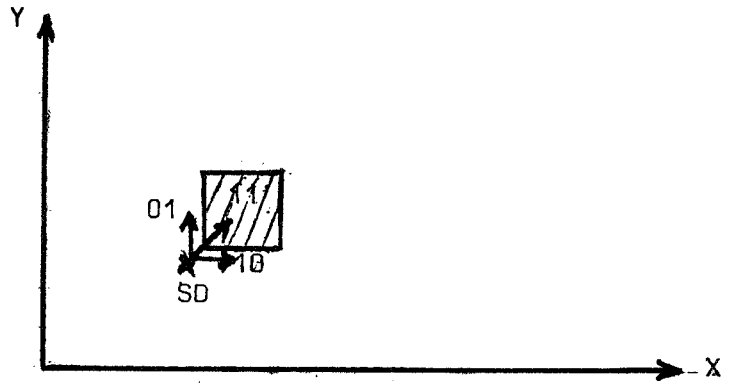
- la connaissance des n variables formant les C_n^2 plans d'analyse,
- les coordonnées des points fournissant les contraintes de chaque plan d'analyse,
- les coordonnées actuelles, (au pas considéré) du point décrivant la trajectoire : soit SD ce point.

L'opérateur analyse est constitué par un certain nombre de modules.

4-1-1- Détection de voisinage d'une contrainte DVC :

Ce module est le premier à entrer en action lors de l'étude dans les C_n^2 plans pour construire un nouveau pas de la trajectoire spatiale. Il détermine les interdictions de couples d'actions du type 10 (X avance, Y arrêté), 01 (X arrêté, Y avance) et 11 (X et Y avancent), c'est-à-dire qu'il mémorise pour chaque plan les couples d'actions interdits (qui pénétreraient dans les contraintes).

Soient x_{SD} et y_{SD} les coordonnées de SD. Ce module détermine ceux des points de coordonnées $x_{SD}+1, y_{SD}$ ou x_{SD}, $y_{SD}+1$ ou $x_{SD}+1$, $y_{SD}+1$ qui sont interdits.



Dans l'exemple de la figure, on constatera que le couple d'actions 11 est interdit (point de coordonnées $x_{SD}+1, y_{SD}+1$).

Remarques : Ce module est ici câblé alors que sur l'optimateur CYBCO, cette partie est programmée avec la recombinaison grâce au tableau INC(I) qui tient compte des progressions interdites dues à la présence de contraintes (3.4.1.).

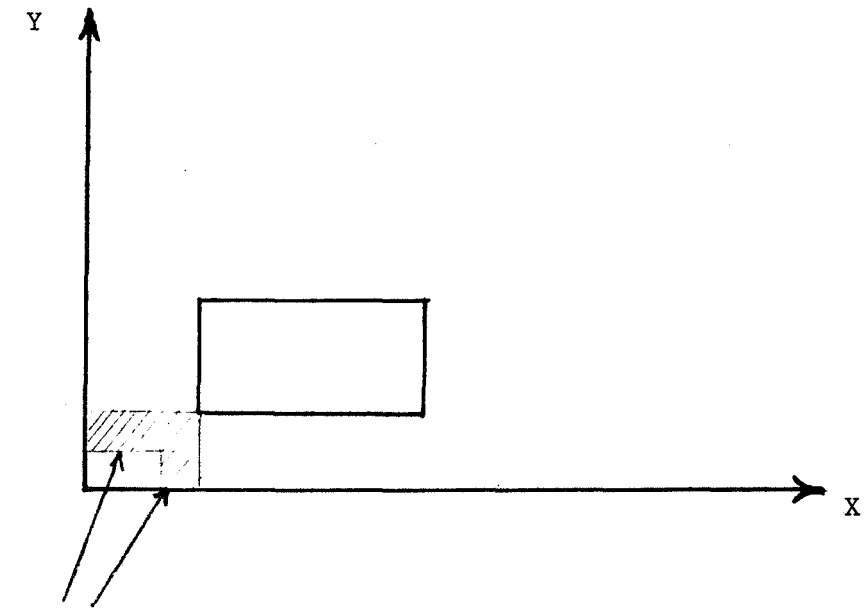
Remarquons ainsi qu'avec le combinateur, la construction du chemin se fait depuis le point figuratif du début de l'ordonnancement jusqu'à la fin alors qu'avec l'optimateur, c'est le contraire.

4-1-2- *Progression S rapide sélective PSRS* :

Son but est de limiter l'exploration des différents plans d'analyse aux seuls points décisifs pour la stratégie globale. Ce module permet ainsi d'éviter de nombreuses explorations.

Les points de décisions sont :

- les points pour lesquels on a détecté une contrainte de voisinage, ce sont les points pour lesquels au moins un couple d'actions a été interdit par le module précédent,
- le premier point SD, point de situation de départ du problème ,
- les points situés sur les deux zones frontières des contraintes.



D'après les résultats obtenus par ce module, les n variables du problème sont classées en deux sous-ensembles disjoints EF et $\overline{EF}$ en désignant par :

. EF les variables en conflit ayant fait l'objet d'une détection de frontière ou ayant des contraintes de voisinage différentes du pas précédent ou étant en situation de début de problème.

. $\overline{EF}$ les variables n'ayant pas fait l'objet d'une détection de frontière, n'ayant pas de contraintes de voisinage sauf si elles sont identiques au pas précédent, et n'étant pas en situation de début de problème.

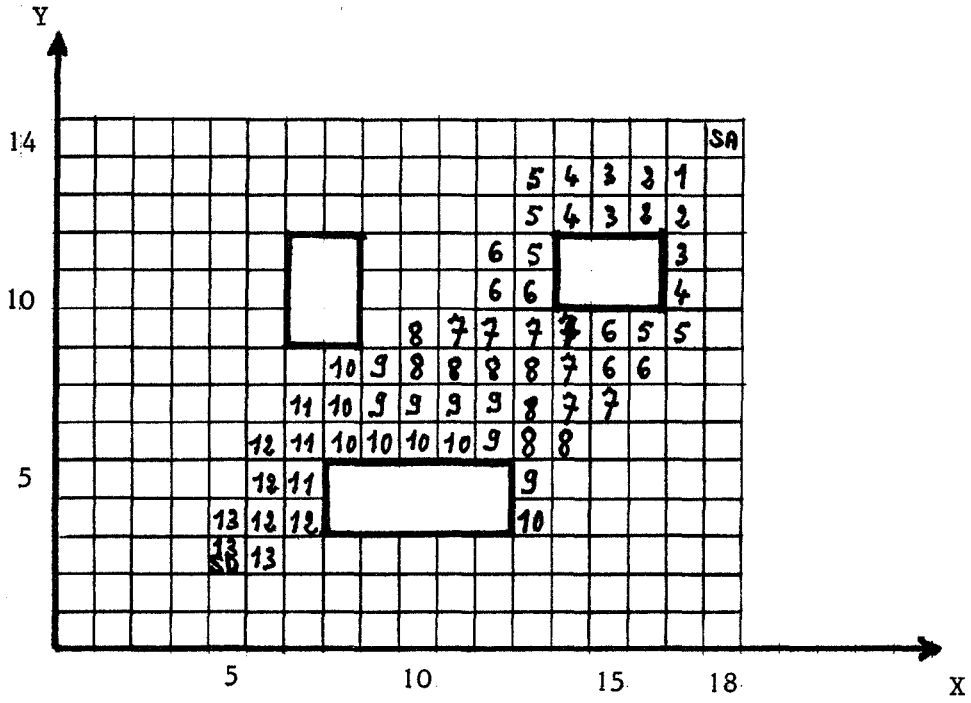
Seules les variables du groupe EF seront soumises à une exploration dans leurs plans, soit k leur nombre. Les variables du groupe $\overline{EF}$ progresseront nécessairement d'un pas, excepté si elles sont soumises aux mêmes contraintes qu'au pas précédent, auquel cas elles effectueront des actions identiques à celles effectuées au cours de ce pas précédent.

4-1-3- Exploration du plan :

L'exploration plane est faite seulement pour les C_K^2 plans correspondant aux K variables en conflit, c'est-à-dire les variables du groupe EF déterminé par la PSRS.

La méthode utilisée est celle de la propagation d'une onde qui est l'image des changements d'états successifs de bascules électroniques reliées les unes aux autres selon une architecture de réseau à mailles régulières, cette onde étant perturbée par les différents obstacles rencontrés.

Considérons un point SA de coordonnées (18,14) dans ce réseau, nous nous proposons d'atteindre le point SD de coordonnées (5,3) à l'aide d'impulsions successives, formant une onde à partir de SA.



Dans l'exemple traité ci-dessus, 13 impulsions auront été nécessaires pour atteindre le point SD.

Cette exploration nous fournit les valeurs des 4 paramètres relatifs au plan (X,Y) dont nous aurons besoin pour la recherche de la solution dans l'espace n-dimensionnel :

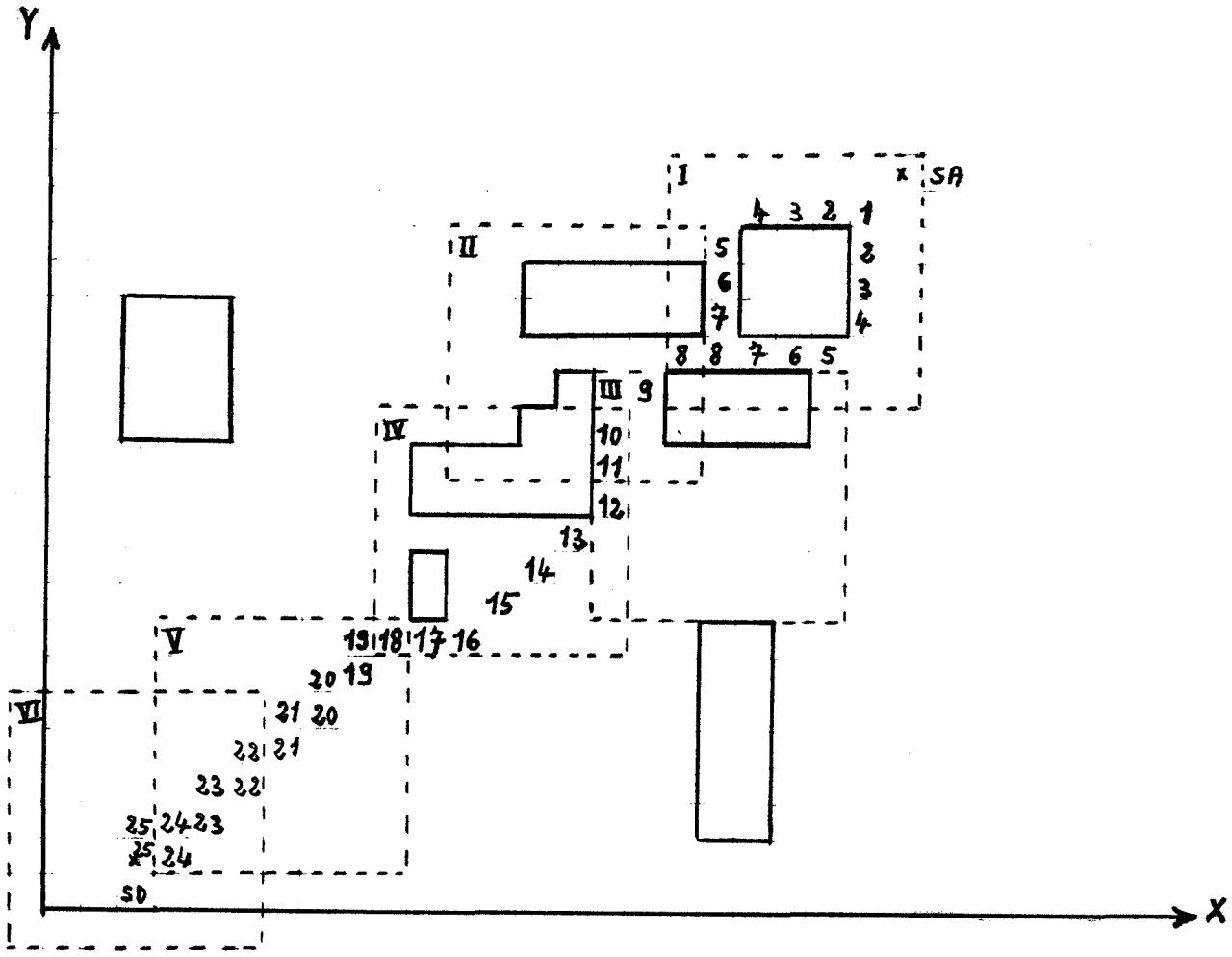
- X action 1, Y action 1 NEXY11 = 12
- X action 1, Y action 0 NEXY10 = 13
- X action 0, Y action 1 NEXY01 = 13
- X action 0, Y action 0 NEXY00 = 13

On peut facilement imaginer des problèmes à traiter qui ont un nombre de pas très important . Techniquement et financièrement, il aurait été difficile de réaliser un réseau maillé de très grande dimension. C'est pourquoi il s'est avéré plus intéressant de construire un réseau maillé carré de dimension limitée, appelé "module de scrutation", et de déplacer ce module dans les différentes parties utiles du plan, de "scruter" le plan.

Cette scrutation peut être soit optimisée, soit diagonale.

- Scrutation optimisée :

Le module se déplace à partir du point SA, point représentatif de la fin de l'évolution des deux processus X et Y, jusqu'au point SD, point représentatif du début de l'évolution des deux processus, de façon à ce que toutes les mailles du réseau par lesquelles passe la trajectoire optimale, soient parcourues.



Algorithme :

Soit L la longueur de l'arête du module de scrutation.

1 - Position I du module :
- angle supérieur droit : point SA de coordonnées (x_{SA}, y_{SA}),
- angle supérieur gauche : point de coordonnées $(x_{SA}-L+1, y_{SA})$,
- angle inférieur droit : point de coordonnées $(x_{SA}, y_{SA}-L+1)$,
- angle inférieur gauche : point de coordonnées $(x_{SA}-L+1, y_{SA}-L+1)$,

2 - Si le point SD est couvert par le module, la scrutation est terminée, sinon : définir une ou deux nouvelles positions du module à partir des bordures verticale gauche et horizontale basse de la position précédente du module en englobant les points atteints en un minimum d'impulsions, retour au début de 2.

Dans le cas représenté sur la figure précédente, le module de scrutation parcourt le plan d'analyse en prenant successivement les positions I à VI afin d'explorer la partie utile du plan, c'est-à-dire celle où figurera le trajet SA - SD optimal. La position i+1 du module se définit par rapport à sa position i par le nombre d'impulsions du front d'onde nécessaires pour atteindre la bordure verticale gauche et horizontale basse du module en position. Les points des bordures atteints avec le minimum d'impulsions seront les points susceptibles de faire partie du chemin optimal, ce qui permet de repositionner le module en position i+1 sur les points afin de continuer à parcourir le chemin optimal. Lorsque le point SD est atteint, la scrutation du plan est terminée, le module en position VI nous donne les valeurs des paramètres.

NEXY11 = 24 NEXY10 = 24

NEXY01 = 25 NEXY00 = 25

- Scrutation diagonale :

Cette méthode s'avère assez efficace lorsque, dans le plan d'analyse, la zone des contraintes se trouve dans une bande comprise entre deux parallèles à la diagonale, suffisamment rapprochées. Les résultats obtenus avec cette méthode ou avec la précédente sont équivalents, seule la disposition des contraintes dans le plan incite à choisir l'une plutôt que l'autre afin de gagner du temps lors de la phase d'exploration du plan d'analyse.

Algorithme :

1 - Définition de la bande diagonale :

- diagonale supérieure : son adresse est la plus haute des adresses hautes des différentes contraintes,
- diagonale inférieure : son adresse est la plus basse des adresses basses des différentes contraintes.

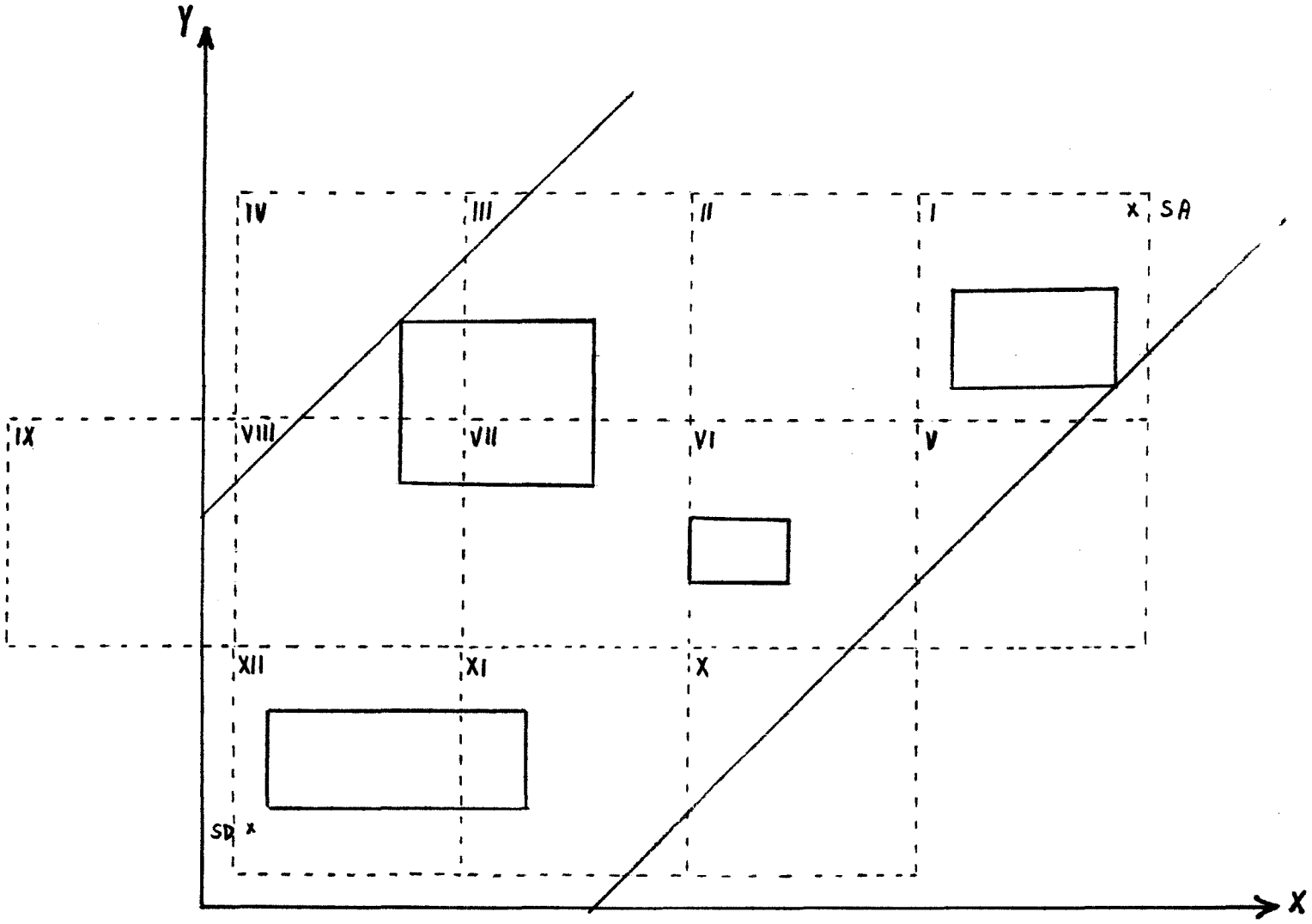
2 - Position I du module :

- angle supérieur droit : point SA de coordonnées (x_{SA}, y_{SA}),
- angle supérieur gauche : point de coordonnées $(x_{SA}-L+1, y_{SA})$,
- angle inférieur droit : point de coordonnées $(x_{SA}, y_{SA}-L+1)$,
- angle inférieur gauche : point de coordonnées $(x_{SA}-L+1, y_{SA}-L+1)$

- i ← 1.

3 - Scrutation, par déplacements successifs du module de la droite vers la gauche, de tous les points d'ordonnée comprise entre $y_{SA}-(i-1)\ L$ et $y_{SA}-iL+1$ situés dans la bande diagonale définie en 1.

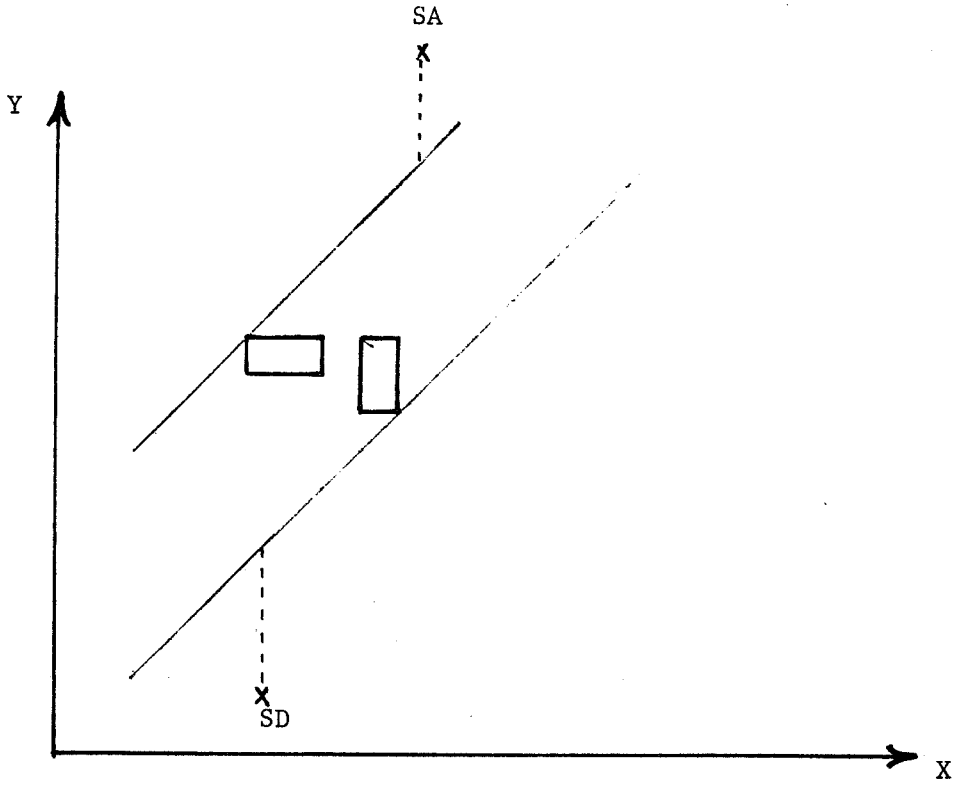
4 - Si le point SD est scruté, le problème est terminé,
sinon : $i \leftarrow i+1$ et retour en 3.



Le premier travail consiste donc à tracer deux diagonales englobant l'ensemble des contraintes afin de limiter l'étude du plan aux zones perturbées. Le module de scrutation prend successivement les positions I à XII afin de parcourir complètement l'intervalle limité par les diagonales et les points SA et SD. Le module s'arrête lorsque le point SD est atteint (position XII) et fournit les valeurs des 4 paramètres NEXY11, NEXY01, NEXY10 et NEXY00.

Remarquons que les découpages du plan déterminant les positions du module de scrutation sont identiques pendant toute l'évolution du point SD, seul le nombre de ces scrutations diminue au fur et à mesure que SD se rapproche de SA.

Si les points SA et SD sont à l'extérieur de la bande définie par les 2 diagonales, l'exploration s'effectue de la même façon que précédemment, il suffit d'ajouter au résultat le nombre de pas nécessaires pour atteindre les diagonales.



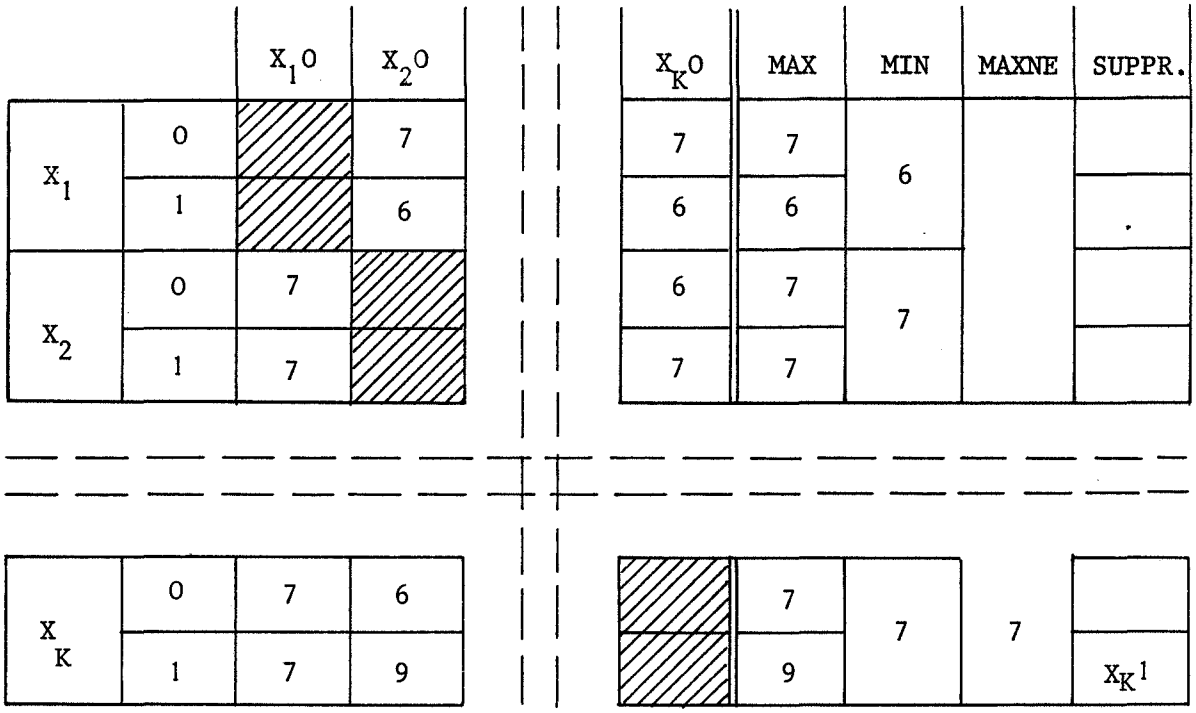
4-2. *L'opérateur recombinaison* :

Les 7 centres ci-après participent à la recombinaison. Ce sont :

- le centre de suppression des actions "CSA",
- le centre d'exploitation des résultats d'analyse "CEA",
- le centre de prétraitement des variables et des actions "CPVA",
- le centre de lever d'ambigüité "CLA",
- le centre d'orientation de la trajectoire "COT",
- le centre de classements intervariables et interactions "CIVCIA",
- le centre de définition du point-situation S(i) "CDS".

4-2-1. *Le centre de suppression des actions "CSA"*:

Ce module a pour rôle de préparer la construction de la trajectoire spatiale T à partir des 4 paramètres, obtenus dans chacun des C_K^2 plans analysés, en supprimant un certain nombre d'actions correspondant aux paramètres les plus élevés.



Pour cela, représentons dans un tableau, à gauche, le coût en nombre de pas, de chaque action de chaque variable prise avec toutes les autres actions 0 des autres variables. Autrement dit, on considère pour chacun des C_K^2 plans d'analyse les coûts de chaque action prise isolément sans associer la progression de l'autre variable du plan examiné.

Exemple :

Dans le tableau ci-dessus, apparaît un coût de 7 pour l'action 0 de X_2 associée à l'action 0 de X_1 dans le plan X_1X_2 et de 7 pour l'action 1 de X_2 associée à l'action 0 de X_K dans le plan X_2X_K, c'est-à-dire $NEX_1X_200 = 7$ et $NEX_2X_K10 = 7$.

Inscrivons dans la colonne MAX le coût maximum de chaque action de chaque variable associée aux actions 0 des autres variables, ce qui revient à inscrire dans MAX le maximum des valeurs de chaque ligne. Dans la colonne suivante, notée MIN, figure la valeur la plus basse des 2 maxima correspondant aux actions 0 et 1 de chaque variable. La colonne MAXNE comporte une seule valeur, la valeur la plus grande figurant dans la colonne MIN. Ainsi, la valeur de MAXNE est le coût minimal pour lequel on a pour chaque variable au moins une action de coût au plus égal à ce coût minimal MAXNE.

Nous décidons de supprimer les actions dont le MAX est supérieur à MAXNE, nous noterons ces suppressions dans la dernière colonne. La méthode employée permet d'éliminer au plus une action par variable.

Algorithme

Soit . $E = \{X_i,\ 1 \leq i \leq K\}$ l'ensemble des variables,

. NEX_iX_j00, $(i \neq j)$, le coût de l'action 0 de la variable X_i associée à l'action 0 de la variable X_j,

. NEX_iX_j10, $(i \neq j)$, le coût de l'action 1 de la variable X_i associée à l'action 0 de la variable X_j.

1 - Déterminer tous les coûts NEX_iX_j00 et NEX_iX_j10 pour $1 \leq i \leq K$, $1 \leq j \leq K$ avec $i \neq j$.

2 - Déterminer pour chaque valeur de i :

$$- MAXO_i = \sup_{\substack{j=1 \\ j \neq i}}^{j=K} (NEX_iX_j00)$$

$$- MAX1_i = \sup_{\substack{j=1 \\ j \neq i}}^{j=K} (NEX_iX_j10)$$

3 - Déterminer pour chaque valeur de i :

$$MIN_i = INF(MAXO_i, MAX1_i)$$

4 - Déterminer $MAXNE = \sup_{i=1}^{i=K} (MIN_i)$

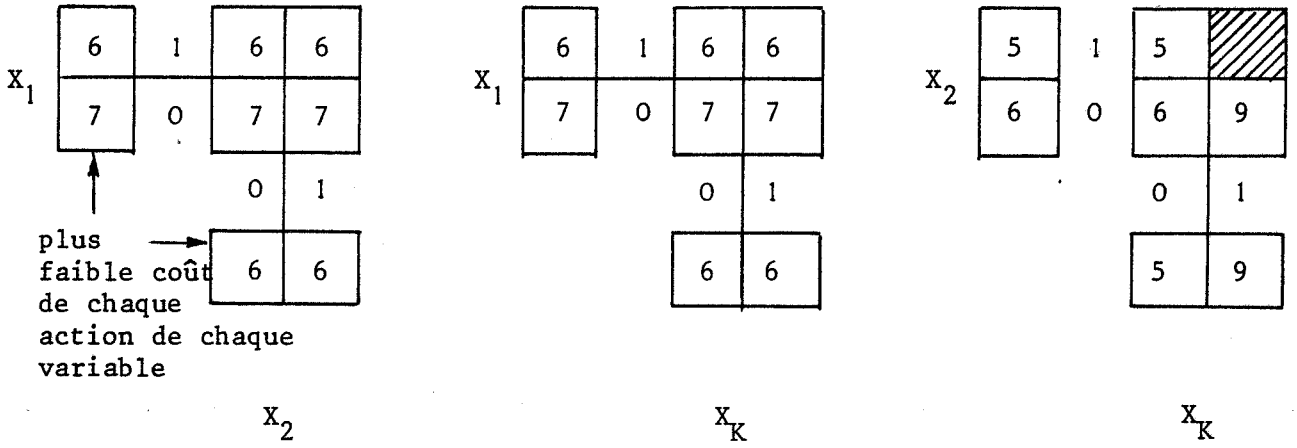
5 - Supprimer tous les couples d'actions suivants :

- action 0 de la variable X_i associée à l'action 0 de la variable X_j dont le coût NEX_iX_j00 est tel que $NEX_iX_j00 > MAXNE$, $1 \leq i \leq K$ et $1 \leq j \leq K$.

- action 1 de la variable X_i associée à l'action 0 de la variable X_j dont le coût NEX_iX_j10 est tel que $NEX_iX_j10 > MAXNE$, $1 \leq i \leq K$ et $1 \leq j \leq K$.

4-2-2- *Le centre d'exploitation des résultats d'analyse : "CEA".*

Ce module calcule, à partir des 4 paramètres obtenus pour chacun des C_K^2 plans d'analyse, le coût minimum de chaque action de chaque variable, associée à l'une ou l'autre des actions de l'autre variable du plan d'analyse, ceci en vue d'élaborer un classement entre les variables et les actions de ces variables, pour sélectionner les actions de plus faible coût.



Exemple :

Nous avons représenté ici l'étude pour les plans d'analyse des 3 variables en conflit X_1, X_2 et X_K. Les matrices 2x2 représentent les coûts de chaque action d'une variable prise avec chacune des 2 actions de l'autre variable, par exemple NEX_2X_K10 coûte 5, NEX_2X_K01 coûte 9 et NEX_2X_K11 est impossible (pénétration dans une contrainte). Les coûts minima de chaque action sont donc les minima de chaque ligne et de chaque colonne. NEX_2OX_K sera le minimum de l'action 0 de X_2 prise avec l'une ou l'autre des actions X_K, ce sera ici le minimum de 6 et de 9 que l'on inscrira dans la case correspondante.

Nous noterons :

$$NEX_iOX_j = INF(NEX_iX_j00, NEX_iX_j01)$$

$$NEX_i1X_j = INF(NEX_iX_j10, NEX_iX_j11)$$

4-2-3- *Le centre de prétraitement des variables et des actions : "CPVA".*

Ce module propose un classement, en fonction des coûts croissants, des K variables entre elles et des deux actions de chaque variable.

		X_1	X_2	X_K	TOT	NE	NEQ	Ambig. variables	Class^t variables	Ambig. actions.	Class^t actions
X_1	0		7	7	7	14	14	2			2
	1		6	6	6	12	12		1		1
X_2	0	6		6	6	12	12	1		1	1
	1	6		5	6	11	6		1	2	1
X_K	0	6	5		6	11	6				1
	1	6	9		9	15	9		3		2

Les premières colonnes indiquent les valeurs des plus faibles coûts pour chaque action de chaque variable X_i prise avec les deux actions d'une autre variable X_j(NEX_i0X_j, NEX_i1X_j) calculées par le module précédent. La colonne TOT est constituée par le coût maximal de chaque action de chaque variable ; ce coût représente le temps minimal au bout duquel la variable X_i aura atteint tous ses points d'arrivée pour l'action correspondante de la ligne. La colonne NE est établie en prenant la somme des coûts pour chaque action de chaque variable pour tous les plans d'analyse. La colonne NEQ enregistre la somme des coûts pour chaque action de chaque variable pour les seules actions dont le coût est égal au maximum (colonne TOT).

La colonne "classement des variables" hiérarchise les variables en fonction des coûts croissants donnés dans la colonne TOT pour les actions 1. Si une ambiguïté subsiste, on tentera de la lever en fonction des coûts pour les actions 0 dans la colonne TOT.

La colonne "classement des actions" hiérarchise les actions de chaque variable en fonction des coûts croissants donnés dans la colonne TOT pour les actions 0 et 1.

Remarquons que les actions 0 et 1 de X_2 n'ont pu être ordonnées l'une par rapport à l'autre. Ce cas d'ambiguïté sera résolu par le module suivant.

Algorithme

Soient $NEX_i.OX_j$ et $NEX_i.1X_j$ calculés par le module précédent.

1 - Déterminer pour chaque variable X_i :

$$\cdot\ TOTO_i = \sup_{\substack{j=1 \\ j\neq i}}^{j=K} (NEX_i.OX_j)$$

$$\cdot\ TOT1_i = \sup_{\substack{j=1 \\ j\neq i}}^{j=K} (NEX_i.1X_j)$$

2 - Déterminer pour chaque variable X_i :

$$\cdot\ NEO_i = \sum_{\substack{j=1 \\ j\neq i}}^{j=K} NEX_i.OX_j$$

$$\cdot\ NE1_i = \sum_{\substack{j=1 \\ j\neq i}}^{j=K} NEX_i.1X_j$$

3 - Déterminer pour chaque variable X_i :

$$\cdot\ NEQO_i = \sum_{\substack{j=1 \\ j\neq i \text{ et } NEX_i.OX_j = TOTO_i}}^{j=K} NEX_i.OX_j$$

$$\cdot\ NEQ1_i = \sum_{\substack{j=1 \\ j\neq i \text{ et } NEX_i.1X_j = TOT1_i}}^{j=K} NEX_i.1X_j$$

4 - Classer les variables X_i en fonction des valeurs croissantes de $TOT1_i$, puis de $TOTO_i$ pour les cas d'égalité, des ambiguïtés peuvent encore subsister.

5 - Classer les actions de chaque variable X_i :

. si $TOTO_i > TOT1_i$, l'action 1 de X_i est classée première, son action 0 seconde,

. si $TOTO_i < TOT1_i$, l'action 1 de X_i est classée seconde, son action 0 première,

. si $TOTO_i = TOT1_i$, l'ambiguïté demeure.

4-2-4- *Le centre de lever d'ambigüité : "CLA".*

Ce module permet de trancher les cas d'ambigüité dans la hiérarchie précédente.

Si une ambigüité de classement subsistait par exemple pour les variables X_1 et X_2, nous considérerions la colonne NE pour les actions 1, pour X_1 on trouve 12 et pour X_2 on trouve 11, X_2 serait donc classée 1 et X_1 serait classée 2, en tenant compte des sommes des différents trajets dans les plans d'analyse. Ceci serait noté dans la colonne "ambigüités des variables" dans le tableau précédent. S'il subsistait encore des ambiguïtés, il faudrait refaire le même travail en considérant la colonne NE pour les actions 0.

Pour lever l'ambigüité entre les actions 0 et 1, nous classons d'abord l'action pour laquelle la différence NE - NEQ est la plus faible, ce qui permet d'éliminer les coûts égaux au maximum de chaque colonne. La différence est de 0 pour l'action 0 de X_2 et de 5 pour l'action 1 de X_2, $X_2 0$ a donc la priorité 1 et $X_2 1$ la priorité 2, ceci est noté dans la colonne "ambigüité des actions" dans le tableau précédent.

Algorithme :

. Cas d'ambigüité des variables :

1 - Classement selon les valeurs croissantes de $NE1_i$,

2 - Si l'ambigüité subsiste, classement selon les valeurs croissantes de NEO_i.

. Cas d'ambigüité des actions :

Soit $\Delta 1_i = NE1_i - NEQ1_i$

et $\Delta O_i = NEO_i - NEQO_i$

- si $\Delta O_i > \Delta 1_i$, l'action 1 de X_i est classée première, son action 0 seconde.

- si $\Delta O_i < \Delta 1_i$, l'action 1 de X_i est classée seconde, son

action 0 première,

- si $\Delta 0_i = \Delta 1_i$, l'ambigüité subsiste.

4-2-5- *Le centre d'orientation de la trajectoire : "COT".*

Ce dernier module est utilisé pour lever les dernières ambiguïtés restant après utilisation du CLA.

Il compare les coûts des actions 0 ou 1 de chaque variable prises avec les actions 1 ou 0 des autres variables par le critère de sommation des coûts des couples d'actions des types 01 et 10.

		X_1		X_2		X_K		NEX_i01	NEX_i10
		0	1	0	1	0	1		
X_1	0				7		7	14	
	1			6		6			12
X_2	0		6				9	15	
	1	7				5			12
X_K	0		6		5			11	
	1	7		9					16
Class^t	Actions	2	1	2	1	1	2		
	variabl.	2		3		1			

Notation :

$$\cdot\ NEX_i01 = \sum_{\substack{j=1 \\ j\neq i}}^{j=K} NEX_iX_j01$$

$$\cdot\ NEX_i10 = \sum_{\substack{j=1 \\ j\neq i}}^{j=K} NEX_iX_j10$$

Les variables ambigües sont classées selon les valeurs croissantes de NEX_i01, ici si X_1, X_2 et X_K n'étaient pas encore classées, elles le seraient dans l'ordre X_K, X_1, X_2.

Les actions ambigües sont classées selon les valeurs relatives de NEX_i01 et NEX_i10. Comme $NEX_1 10 < NEX_1 01$, l'action 1 de la variable X_1 serait classée avant son action 0, si on avait $NEX_1 10 > NEX_1 01$, l'action 1 serait classée après l'action 0.

Remarques :

Les actions impossibles (pénétration d'une contrainte) sont pénalisées d'un coût infini.

La recombinaison décrite ci-dessus peut selon la nature des problèmes être légèrement modifiée suivant la précision désirée des résultats et les temps de calcul autorisés.

Si, après l'étude précédente, des ambigüités subsistent dans le classement, ce qui signifie qu'il existe plusieurs trajectoires équivalentes, la décision sera prise en dernier ressort par l'opérateur.

4-2-6- *Le centre de classements intervariables et interactions : "CIVCIA".*

Ce module est simplement un tableau regroupant les classements des variables et des actions des différentes variables.

Nous remarquons sur ce tableau l'interdiction de l'action 1 de la variable X_K, décidée par le centre de suppression des actions CSA.

		X_1	X_2	X_K
Classement des variables		2	1	3
Classement des actions	0	2	2	1
	1	1	1	2

4-2-7- Le centre de définition du point-situation S(i') : "CDS".

Ce module a pour fonction de définir un pas de la trajectoire spatiale en proposant un point-situation S(i') de l'espace. Les coordonnées de S(i') pour les variables X_1, X_2, ..., X_K seront déterminées par le tableau du paragraphe 4-2-6 en prenant les actions classées premières pour chacune des variables, ces coordonnées sont notées $S(i')X_1$, $S(i')X_2$, ..., $S(i')X_K$. Afin de vérifier si S(i') est compatible avec les contraintes, nous interrogeons le tableau du paragraphe 4-2-2 dans lequel figurent tous les couples d'actions interdits, c'est-à-dire des actions pour lesquelles la trajectoire pénétrerait dans une contrainte.

Si S(i') est compatible avec les contraintes, ses coordonnées relatives aux variables X_1, X_2, ...,X_K sont déterminées, les coordonnées relatives aux N - K variables non en conflit X_{K+1}, X_{K+2}, ..., X_N du sous-ensemble $\overline{EF}$ seront déterminées en reconduisant, pour ces variables, les actions du pas précédent (voir paragraphe 4-2-1).

Si S(i') n'est pas compatible avec les contraintes, nous proposons un nouveau point-situation S(i') en changeant l'action de la variable classée en dernière position et nous vérifions de nouveau la compatibilité. Si le test est négatif, il faut réitérer la méthode.

Cette méthode, qui consiste à former successivement des jeux de coordonnées candidats à la définition du point S(i'), permet ainsi de tester en premier lieu, la compatibilité sur les jeux de plus faibles coûts, donc sur ceux qui ont la plus forte probabilité de conduire à une trajectoire spatiale T, optimale.

. Exemple

Considérons le cas des 3 variables X_1, X_2, X_K qui figurent au tableau du paragraphe 4-2-6. Les actions proposées permettant de définir S(i') sont les suivantes :

- variable classée première : X_2, action 1,
- variable classée seconde : X_1, action 1,
- variable classée troisième : X_K, action 0.

Le tableau du paragraphe 4-2-2 nous indique que les actions 1 simultanées de X_1 et X_2 sont impossibles car la trajectoire pénètrerait dans une contrainte. Il faut donc proposer un second point S(i') en changeant l'action de la variable classée troisième. Or l'action 1 de X_K est interdite, on va donc changer l'action de la variable classée seconde, c'est-à-dire proposer l'action 0 de X_1,

ce qui conduit à l'ensemble d'actions suivant :

- action 1 pour X_2,
- action 0 pour X_1,
- action 0 pour X_K.

Cet ensemble d'actions est compatible avec le tableau du paragraphe 4-2-2 et permet de calculer les coordonnées $S(i')X_1$, $S(i')X_2$, ..., $S(i')X_K$ à partir des coordonnées du point $S(i'-1)$, il suffit maintenant d'ajouter à cet ensemble de coordonnées, l'ensemble des coordonnées des N-K variables non en conflit, le point $S(i')$ sera complètement déterminé. Le vecteur $\overrightarrow{S(i'-1)S(i')}$, de module l'unité, définit le pas de la trajectoire. Pour définir le point $S(i'+1)$, il faut de nouveau utiliser l'opérateur analyse et reprendre la méthode au paragraphe 4-1.

Si aucun ensemble d'actions n'avait été compatible et qu'il aurait été impossible de définir un point $S(i')$, il aurait fallu augmenter MAXNE (défini au paragraphe 4-2-1) et reprendre les calculs à ce niveau. Cela aurait eu pour effet de diminuer le nombre d'actions supprimées par le CSA. Au cas où le CSA ne supprimerait aucune action et qu'aucun point $S(i')$ ne serait acceptable, le problème devrait être considéré "sans solution", aucune trajectoire ne pouvant être construite pour relier SD à SA. Bien entendu, un test interdit le cas où toutes les variables feraient l'action 0, ce qui conduirait à un point stationnaire de la trajectoire.

. Algorithme.

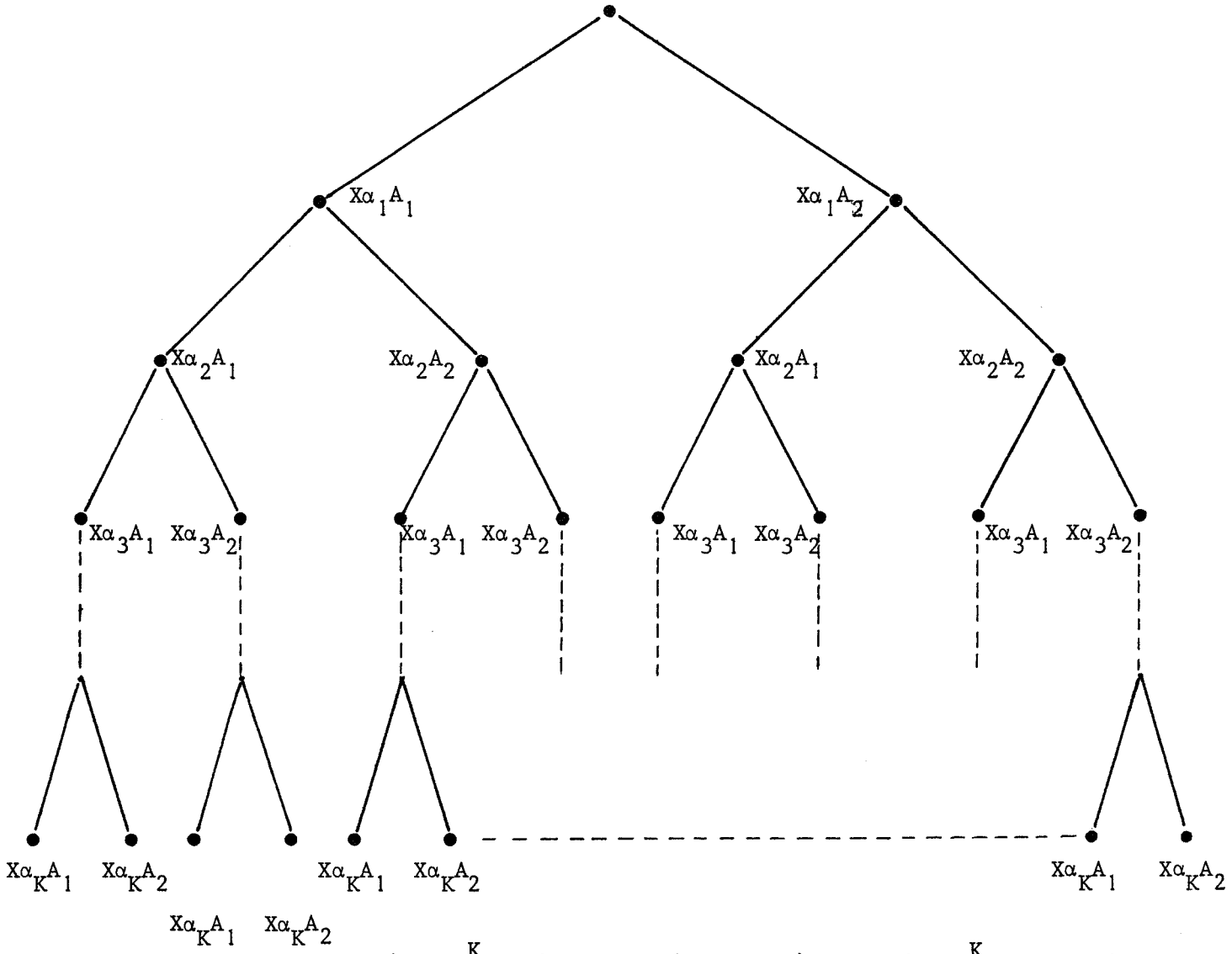
Soit $S(i')$ le point-situation de l'espace de dimension N qu'on se propose de déterminer, on connaît déjà les points $S(0) = SD$, $S(1)$, $S(2)$, ..., $S(i'-1)$.

Soient $S(i')X_1$, $S(i')X_2$, ..., $S(i')X_K$, $S(i')X_{K+1}$, ..., $S(i')X_N$, les coordonnées de $S(i')$.

1 - Obtention du classement des variables et des actions par le "CIVCIA" (paragraphe 4-2-6), soient :

- X_{α_1}, X_{α_2}, ..., X_{α_K} les variables en conflit classées,
- $X_{\alpha_j} A_1$ l'action classée première de la variable X_{α_j},
- $X_{\alpha_j} A_2$ l'action classée seconde de la variable X_{α_j}.

2 - Rangement, dans un arbre, de toutes les actions des K variables en conflit, de la façon suivante :



On obtient 2^K feuilles terminales, c'est-à-dire 2^K permutations possibles des actions des K variables pour définir le point S(i'). Numérotons ces feuilles de la gauche vers la droite, soit m le rang d'une feuille $1 \leqslant m \leqslant 2^K$.

Prenons d'abord m = 1, et considérons la branche aboutissante. Elle détermine les K actions classées premières pour chacune des variables.

3 - Obtention du tableau des incompatibilités (paragraphe 4-2-2).

4 - Etude, en consultant le tableau du paragraphe 4-2-2, de la compatibilité des actions proposées par la branche de rang m pour les K variables en conflit :

- si compatibilité :

$|\underline{a}|$ les coordonnées de S(i') : $S(i')X_1$, $S(i')X_2$, ..., $S(i')X_K$ et $S(i')X_{K+1}$, $S(i')X_{K+2}$, ..., $S(i')X_N$ sont définies à partir de celles de S(i'-1) de la façon suivante :

. si on a décidé l'action 0 pour la variable X_i :

$$X(i')X_i = S(i'-1)X_i$$

. si on a décidé l'action 1 pour la variable X_i :

$$S(i')X_i = S(i'-1)X_i+1$$

|b| Si S(i') = SA, la trajectoire est complètement déterminée, sinon i' ← i'+1, retour en 1.

- si incompatibilité : m ← m + 1,

|a| si $m \leq 2^K$, la nouvelle branche aboutissante fournit-elle un ensemble de K actions acceptables au regard du tableau du paragraphe 4-2-6 ?

- si oui, retour en 4,
- sinon, m ← m + 1, retour au début de a.

|b| si $m > 2^K$, existe-t-il au moins une valeur m_o de m, $(1 \leq m_o \leq 2^K)$, telle que la branche correspondante ait été supprimée au regard du tableau du paragraphe 4-2-6 ?

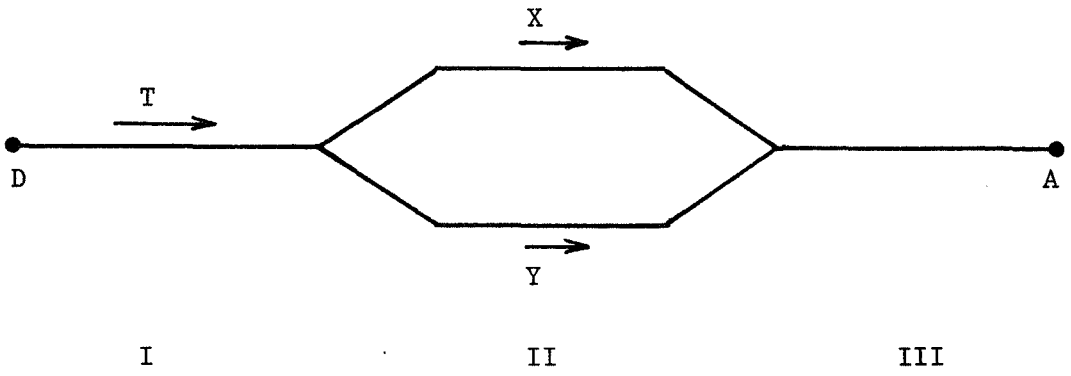
- si oui, MAXNE ← MAXNE + 1, retour en 1,
- sinon, le problème est impossible.

4-3- *Choix de moyens* :

Ce module est une adjonction éventuelle à la machine décrite jusqu'à maintenant ; il étend le champ d'application de combinateur à des classes de problèmes combinatoires dans lesquels interviennent des contraintes dites cumulatives, c'est-à-dire où il existe plusieurs moyens d'un même type qui sont candidats à la réalisation d'une même tâche. C'est le cas en trafic ferroviaire lorsqu'un train a le choix entre plusieurs itinéraires pour atteindre une même destination ou en ordonnancement d'atelier lorsque plusieurs machines-outils identiques peuvent usiner la même pièce.

Selon ce module, les variables du combinateur (évoluant sur les axes de l'espace n-dimensionnel) ne sont plus des mobiles (trains) qui défilent sur des moyens (voies). Les variables représentent au contraire les moyens (voies) qui défilent sous des mobiles (trains) qui sont alors considérés comme fixes.

Une variable est alors affectée à chaque moyen et non à chaque mobile. Il y aura autant de variables que de moyens (voies) candidats au traitement d'un mobile (train).

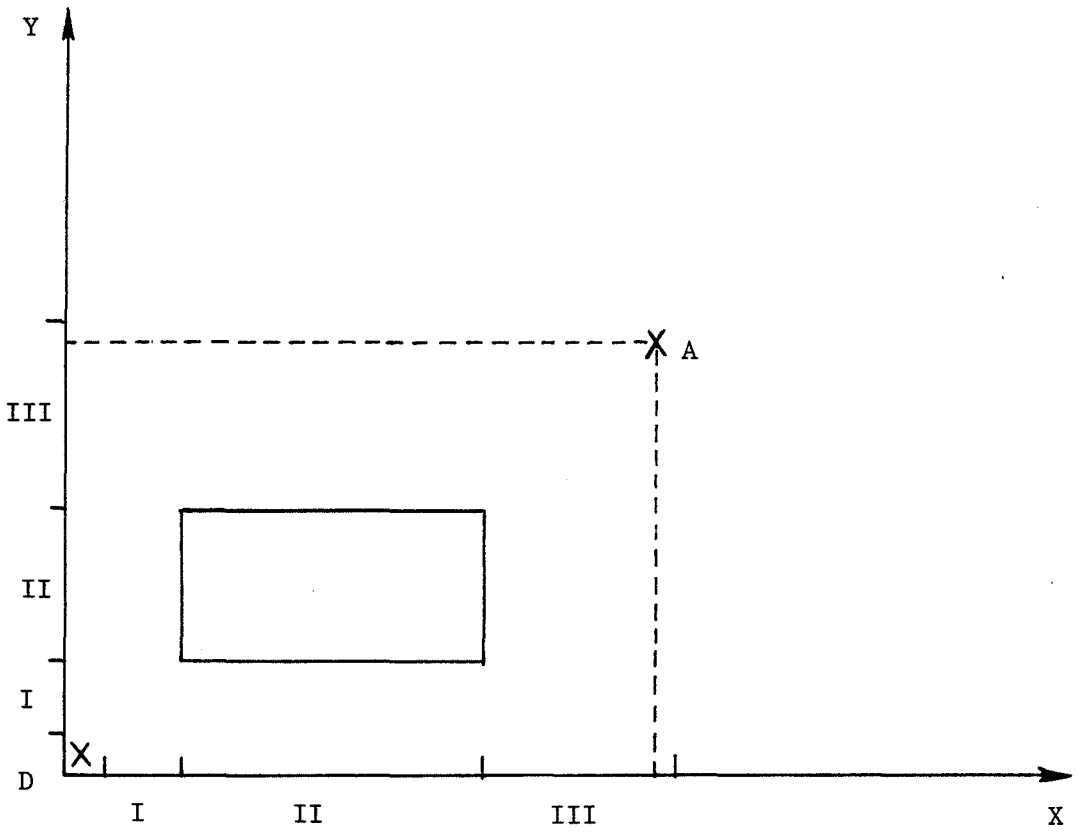


Exemple :

Sur la figure ci-dessus, le train T circulant de D vers A a le choix entre les itinéraires X et Y pour relier le point D au point A.

Nous allons représenter ce problème dans le plan d'analyse. Les axes X et Y correspondront aux trajets X et Y. La contrainte tiendra compte du fait que dans la région II, le train T ne peut occuper simultanément les parcours X et Y (voir page suivante).

Remarquons que lorsque le choix vient d'être réalisé, (contourner la contrainte soit par le haut, soit par le bas ; c'est-à-dire emprunter soit le trajet X, soit le trajet Y), un moyen (voie) traite le mobile (train T) et l'autre moyen (autre voie) n'est pas utilisé et ne progressera plus (dans le plan d'analyse). C'est pourquoi, dès que le choix est fait, la variable qui représente le moyen non utilisé doit disparaître.



Plan d'analyse des 2 itinéraires X et Y

5 - ETUDE COMPARATIVE DES DIVERSES RECOMBINAISONS ET HEURISTIQUES

5 - 1. LES DIFFERENTES METHODES TESTEES

Nous avons résolu différents problèmes d'ordonnancement en utilisant l'analyse bidimensionnelle associée à diverses recombinaisons (majoritaire, arborescente et recombinaison du combinateur) déjà examinées. Nous avons aussi résolu certains de ces problèmes par des méthodes heuristiques purement logicielles et par un apprentissage en vue de comparer les résultats et rechercher les améliorations éventuelles apportées par les calculateurs spécialisés dans le traitement des problèmes combinatoires. Notons dès maintenant que, sauf pour une batterie de problèmes de trains au départ qui a été traitée sur l'"optimateur" les calculateurs spécialisés ont toujours été simulés sur un ordinateur PHILIPS P1175.

5-1-1- *Méthodes utilisant des calculateurs spécialisés*

5-1-1-1- *Utilisation de l'optimateur CYBCO*

Pour traiter les problèmes avec l'optimateur CYBCO, nous avons connecté l'optimateur à un ordinateur Télémécanique T1600 sur lequel a été programmé une première version de l'algorithme majoritaire exposé en 3-4-1. L'algorithme de recombinaison utilisé ici ne prenait pas en compte la différence exacte entre $T_p(A')$ et $T_p(A'')$ pour chaque plan, mais 0 ou 1 selon le signe de la différence, ce qui revenait à augmenter d'une unité la valeur d'un compteur associé au processus qu'il était préférable de faire progresser.

5-1-1-2- *Simulation des calculateurs spécialisés*

Des modèles des différentes recombinaisons ont été simulés sur ordinateur. Bien entendu, ces modèles ne tiennent pas compte des modifications qui ont pu être apportées récemment par les sociétés (SNCF et CYBCO) ayant la charge de développer industriellement les calculateurs spécialisés que nous

avons étudiés. Néanmoins, les simulations réalisées dans le cadre de cette étude utilisent les idées fondamentales des diverses recombinaisons.

Bien qu'utilisant des algorithmes différents pour l'analyse bidimensionnelle, l'optimateur et le combinateur fournissent le même résultat quant au retard minimum dans chaque plan de projection. C'est pourquoi nous avons conçu un programme unique pour l'analyse bidimensionnelle. Par contre, les recombinaisons associées aux deux procédés fournissent des résultats différents.

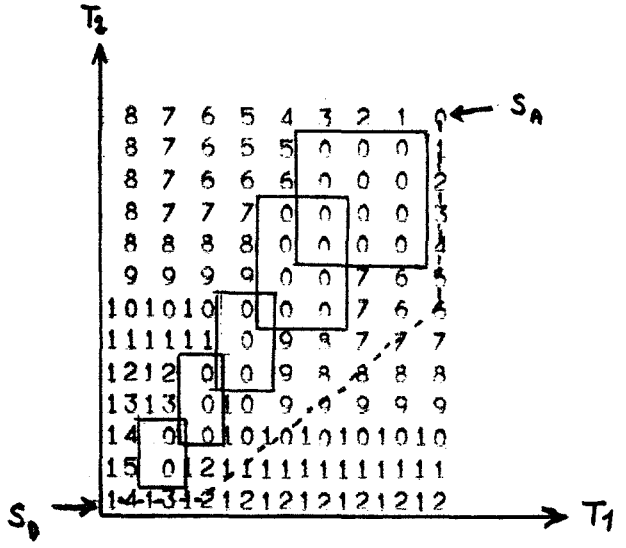
Ce programme simule le déplacement du front d'onde dans chacun des plans de projection parsemés d'obstacles. Les données introduites sont :

- la liste des différents processus,
- la liste des différents moyens (machines dans un atelier, voies de chemin de fer, etc...)
- les gammes avec les temps d'occupation des moyens par les différents processus.

Le programme calcule ensuite les états successifs du front d'onde dans chaque plan et enregistre les résultats sur disque.

Exemple :

PLAN(2, 1)



Nous avons figuré ici un listing représentant l'analyse bidimensionnelle du plan de projection associé aux processus T_1 et T_2, les rectangles remplis de zéros étant les obstacles.

Les états successifs du front d'onde dans chaque plan constituent les données d'un second programme qui simule les trois recombinaisons dont il s'agit de comparer les performances.

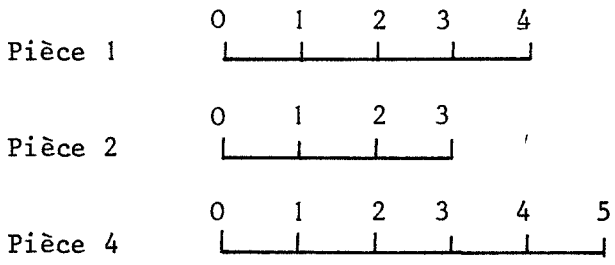
5-1-2- *Méthodes logicielles [13]*

Nous avons programmé trois heuristiques très simples et une méthode par apprentissage qui utilise les trois heuristiques afin de tester les recombinaisons. Ces quatre méthodes ont servi à résoudre des problèmes classiques d'ordonnancements d'ateliers. Il s'agit de réaliser n pièces sur m machines. Pour chaque pièce, on suppose connus la gamme de fabrication et le temps d'occupation de chaque machine. On cherchera à minimiser le délai global de fabrication, c'est-à-dire le temps entre le début du passage de la première pièce sur la première machine et la fin du passage de la dernière pièce sur la dernière machine.

5-1-2-1- *Méthode C. D. (Plus courte durée ou S. I. O. : Shortest Imminent Operation)*

Selon cette méthode, lorsqu'une machine est libre, on y affecte par priorité la pièce, choisie parmi toutes celles en attente de cette machine, dont le temps de passage est le plus court. Supposons par exemple, qu'au cours de la simulation du problème d'atelier la machine 3 vienne de terminer le traitement d'une pièce. Les pièces 1, 2 et 4 sont en attente pour passer sur la machine 3, les temps de passage sur cette machine étant de 4 heures pour la pièce 1, 3 heures pour la pièce 2 et 5 heures pour la pièce 4. La méthode C. D. nous permet d'affecter la pièce 2 à la machine 3 puisque c'est cette pièce qui a le temps de passage le plus court sur cette machine.

Schéma des temps de passage sur la machine 3 :



Le but de cette heuristique est de faire exécuter les travaux les plus courts par priorité, afin de ne pas bloquer une machine trop longtemps avec une tâche trop longue car, pendant ce temps, d'autres machines seraient

susceptibles de se libérer et de ne plus avoir de pièces à traiter. Cette heuristique semble particulièrement bien adaptée au début de l'ordonnancement d'une chaîne car elle permet d'alimenter au plus vite toutes les machines.

5-1-2-2- *Méthode L. D. (Plus longue durée ou L. I. O. : longest imminent operation)*

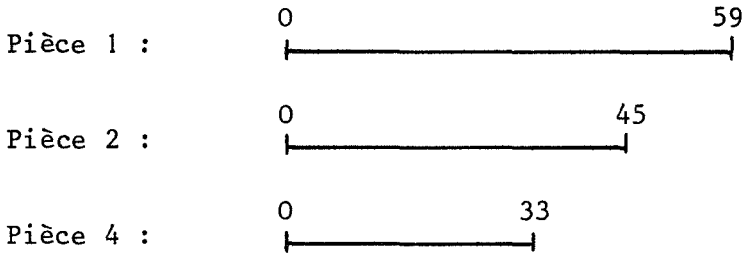
L'heuristique consiste ici à faire passer sur une machine libérée, par priorité la pièce dont le temps de traitement sera le plus long sur cette machine.

En reprenant l'exemple du paragraphe précédent, il apparaît que nous choisirons ici de faire traiter d'abord la pièce 4 par la machine 3.

5-1-2-3- *Méthode L. T. R. (Plus long temps résiduel ou L. R. T. : longest remaining time)*

Lorsqu'une machine est libre, on lui affecte par priorité la pièce en attente de cette machine qui est la plus "critique", c'est-à-dire celle dont le temps total minimum nécessaire à l'exécution complète de la fin de sa gamme est le plus long.

Schéma des temps résiduels :



Supposons comme précédemment, que la machine 3 vienne de terminer le traitement d'une pièce. Les pièces 1, 2 et 4 sont en attente pour passer sur la machine 3, les temps nécessaires à l'exécution complète de la fin de leur gamme étant respectivement de 59 heures, 45 heures et 33 heures. La méthode LTR permet d'affecter la pièce 1 sur la machine 3 puisque c'est cette pièce qui est la plus "critique" parmi les 3.

5-1-2-4- *Méthode par apprentissage*

Cette méthode demande un temps de calcul nettement plus élevé que les heuristiques précédentes, mais conduit en général à des meilleurs ordonnancements. L'apprentissage utilise ici les méthodes C.D., L.D. et L.T.R. et tente d'orienter les différentes étapes de l'ordonnancement en fonction de résultats précédemment obtenus d'où le nom d'apprentissage. Cette méthode est calquée

sur la démarche du cerveau humain qui, lorsqu'il doit prendre une décision, recherche dans son passé s'il ne s'est pas déjà trouvé dans un cas semblable, et si oui, sa décision sera grandement influencée par le fait que ses décisions précédentes se soient soldées par une réussite ou bien par un échec.

Pour traiter un problème d'ordonnancement par apprentissage, nous nous donnons d'abord un certain nombre d'heuristiques habituellement utilisées pour résoudre ce type de problèmes. Nous appliquons ensuite successivement ces différentes heuristiques. Seules, celles qui nous fournissent les meilleures solutions et qui semblent donc les mieux appropriées au problème, seront utilisées dans l'apprentissage.

Appelons décision l'action d'affecter une pièce à une machine. Pour un problème à n pièces et m machines, chaque gamme ayant pour longueur m, nous avons n x m décisions à prendre. Attachons un vecteur probabilité à chaque décision de sorte qu'à la K-ième décision soit attaché un vecteur V_K. Ce vecteur probabilité a autant de composantes qu'il y a d'heuristiques utilisées dans l'apprentissage (3 dans l'exemple considéré), notons $V_K(P_{1K}, P_{2K}, P_{3K})$ ce vecteur avec :

- P_{1K}, probabilité d'utiliser H1 (première heuristique conservée) pour prendre la décision de rang K.

- P_{2K}, probabilité d'utiliser H2 (seconde heuristique conservée) pour prendre la décision de rang K.

- P_{3K}, probabilité d'utiliser H3 (troisième heuristique conservée) pour prendre la décision de rang K.

Initialement $P_{1K} = P_{2K} = P_{3K} = \frac{1}{3}. \forall K$. Appelons "ordonnancement standard", (noté O.S.), le meilleur des 3 ordonnancements obtenus par les heuristiques. Nous déterminons ensuite un nouvel ordonnancement à partir des n x m décisions. L'heuristique appliquée à chaque point de décision est déterminée par le tirage aléatoire d'un nombre X_K compris entre 0 et 1, pour la K-ième décision nous utiliserons :

. H1 si $\quad 0 \leqslant X_K < P_{1K}$

. H2 si $\quad P_{1K} \leqslant X_K < P_{1K} + P_{2K}$

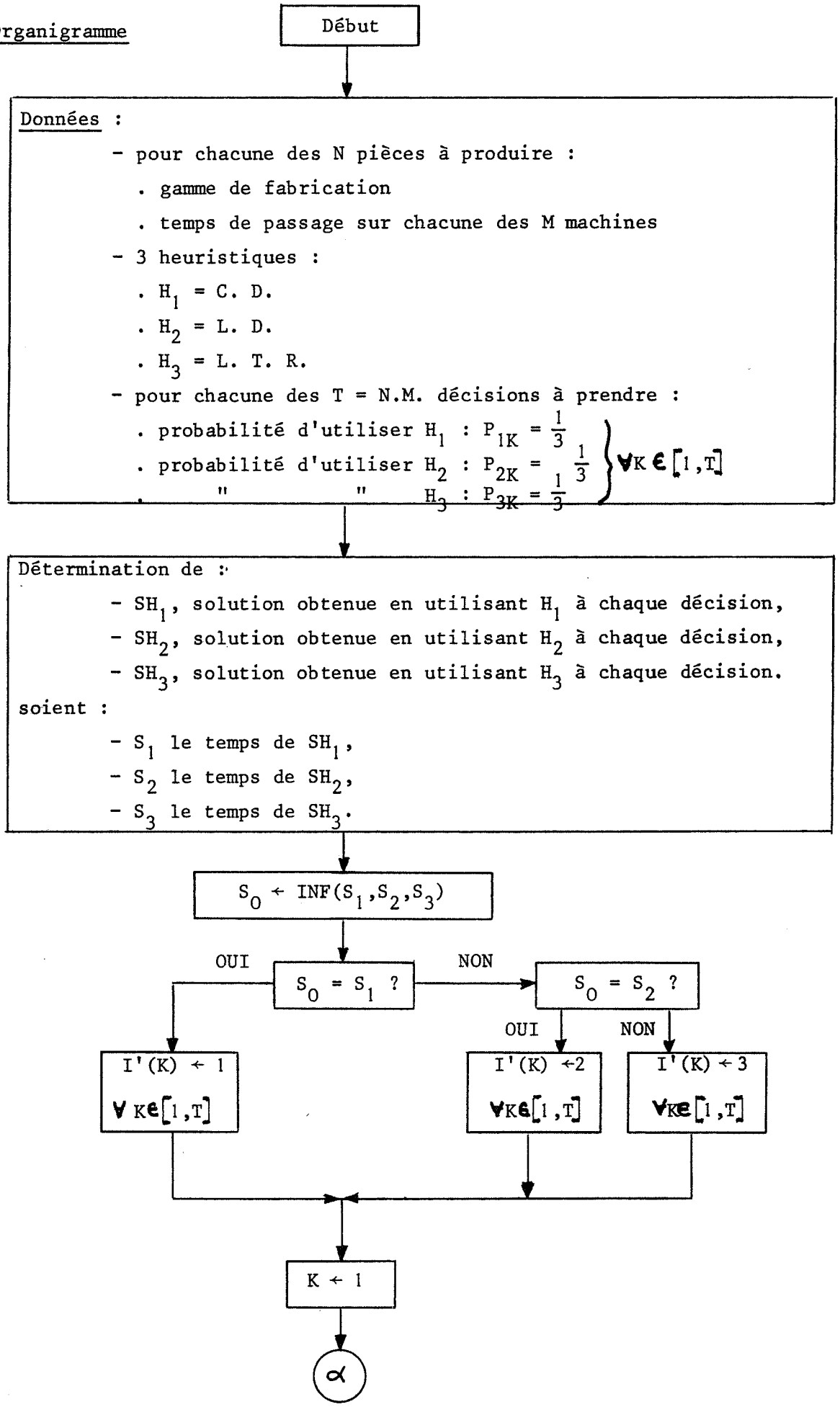
. H3 si $\quad P_{1K} + P_{2K} \leqslant X_K \leqslant 1$

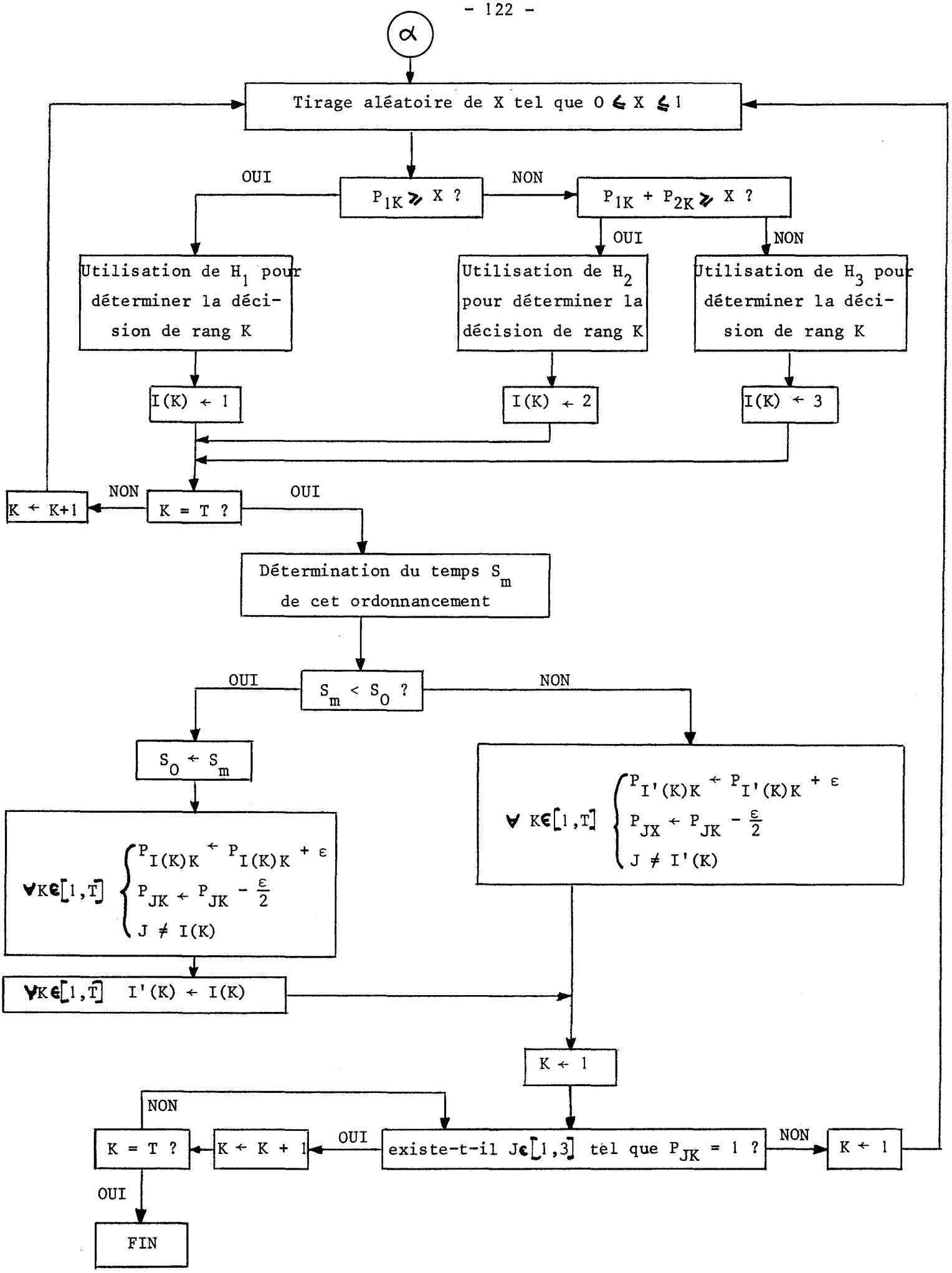
Si le temps de ce dernier ordonnancement est meilleur que celui de O.S., on augmente de ε, pour chacun des n x m vecteurs, la composante ayant contribué à l'obtention de cette solution, et on diminue de $\frac{\varepsilon}{2}$ toutes les autres composantes. Ce dernier ordonnancement devient le nouvel O.S..

Par contre, si cette dernière solution est plus mauvaise que O.S. on augmente de ε les composantes correspondant à O.S. et on diminue de $\frac{\varepsilon}{2}$ les autres.

Nous itérons ensuite la méthode en cherchant un nouvel ordonnancement jusqu'à ce que chaque vecteur V_K converge vers un vecteur limite V'_K.

. Organigramme




α
Tirage aléatoire de X tel que 0 ≤ X ≤ 1
OUI
P1K ≥ X ?
NON
P1K + P2K ≥ X ?
OUI
NON
Utilisation de H1 pour déterminer la décision de rang K
Utilisation de H2 pour déterminer la décision de rang K
Utilisation de H3 pour déterminer la décision de rang K
I(K) ← 1
I(K) ← 2
I(K) ← 3
NON
K ← K+1
K = T ?
OUI
Détermination du temps Sm de cet ordonnancement
OUI
Sm < S0 ?
NON
S0 ← Sm
∀K∈[1,T] { PI(K)K ← PI(K)K + ε ; PJK ← PJK - ε/2 ; J ≠ I(K) }
∀K∈[1,T] I'(K) ← I(K)
∀K∈[1,T] { PI'(K)K ← PI'(K)K + ε ; PJX ← PJK - ε/2 ; J ≠ I'(K) }
K ← 1
NON
K = T ?
K ← K + 1
OUI
existe-t-il J∈[1,3] tel que PJK = 1 ?
NON
K ← 1
OUI
FIN

. Etude la convergence

A chaque simulation, pour toute valeur de K, il existe une valeur $J \in \{1,2,3\}$ pour laquelle P_{JK} augmente de ε et $P_{J'K}$ diminue de $\frac{\varepsilon}{2}$, $\forall$ $J' \neq J$.

Le temps de chaque ordonnancement est borné inférieurement et supérieurement. Au cours des simulations, le temps de l'ordonnancement standard prend des valeurs entières successives décroissantes minorées par le temps de l'ordonnancement optimal inconnu. Nécessairement, après un nombre fini de simulations, l'ordonnancement standard ne varie plus, donc pour chacun des K vecteurs, J devient fixe et par conséquent P_{JK} tend vers 1. L'étude est alors terminée. Pour chaque vecteur V_K, une heuristique est définie, c'est celle correspondant à $P_{JK} = 1$. Les simulations que l'on obtiendrait par la suite seraient toutes identiques à l'ordonnancement standard.

Autre démonstration de la convergence

Soient :

- S_m le temps de l'ordonnancement standard obtenu à la suite de la simulation de rang m.

- S le temps de l'ordonnancement optimal inconnu.

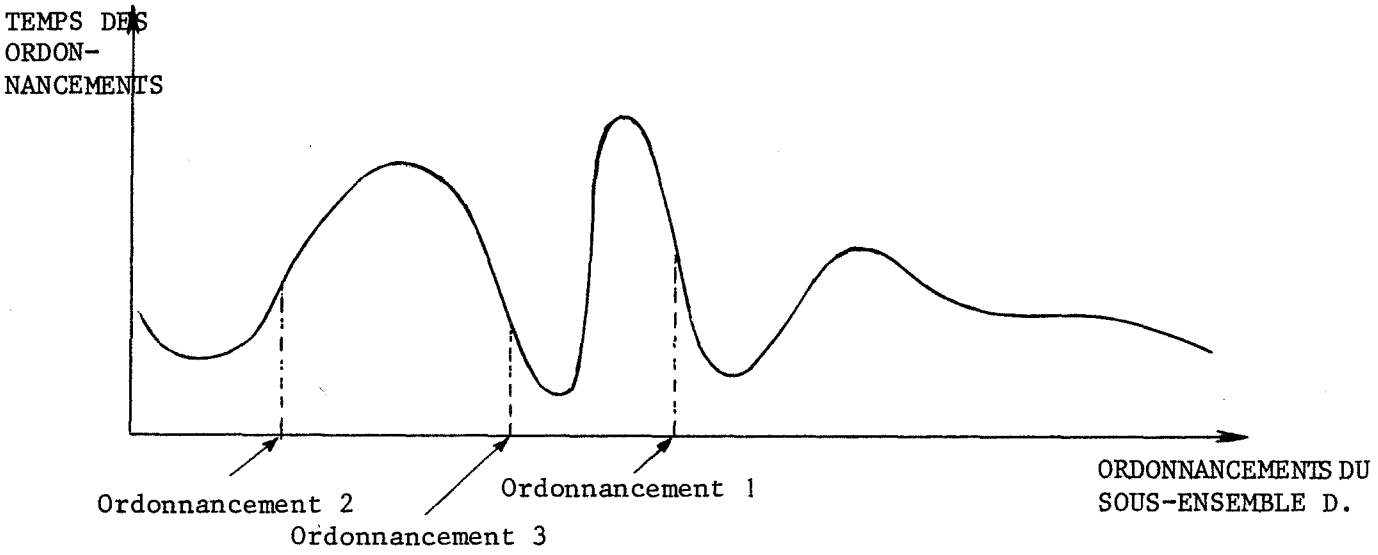
- S_o le temps de l'ordonnancement standard initial.

Quel que soit le rang m, $S_{m+1} \leqslant S_m$ et $S \leqslant S_m$, les termes de la forme S_m forment une suite de nombres entiers décroissante et minorée, donc convergente.

Danger du minimum local

Remarquons que les ordonnancements obtenus par apprentissage appartiennent à un sous-ensemble D de l'ensemble des ordonnancements possibles pour chaque problème. C'est le sous-ensemble constitué des ordonnancements obtenus en appliquant soit C.D., soit L.D., soit L.T.R. à chaque point de décision.

L'apprentissage est susceptible de nous fournir simplement un minimum local du sous-ensemble D des ordonnancements d'un problème. En effet, l'apprentissage est l'art d'utiliser les résultats des actions passées pour préparer les actions futures, ce qui signifie que l'on cherche à améliorer l'ordonnancement standard en testant des ordonnancements assez voisins de ceux qui ont déjà donné de bons résultats, ce qui conduit généralement à un minimum local.



Pour essayer de remédier à cela, les probabilités de choisir l'une des trois heuristiques, égales à chaque point de décision au début du problème, ne varient que lentement pour les premières simulations afin de tenter d'explorer la plupart des régions du sous-ensemble D. Ce n'est qu'après un certain nombre de simulations que nous ferons croître plus rapidement les probabilités de tirage des heuristiques associées à l'ordonnancement standard. Le risque de se trouver dans un minimum local est alors diminué, mais non supprimé.

5 - 2. LES DIFFERENTS PROBLEMES TRAITES

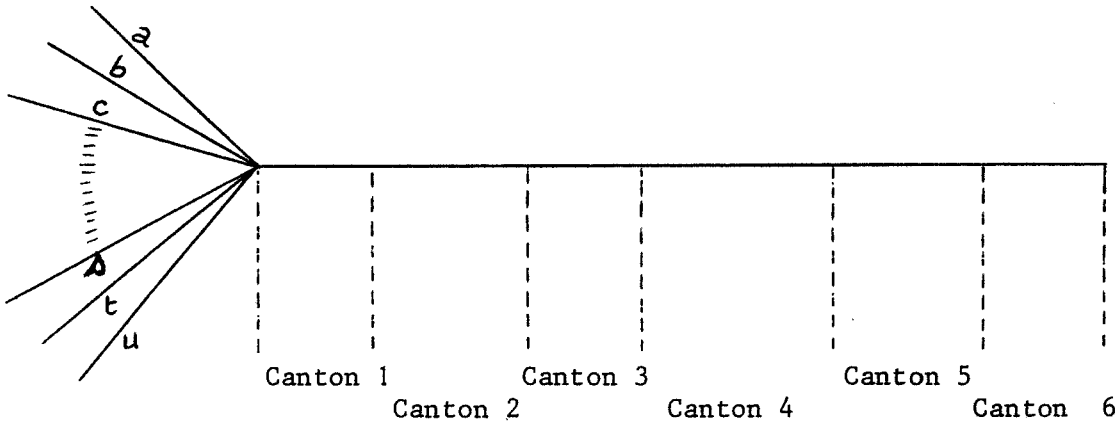
Les cas pratiques sur lesquels a porté l'évaluation des méthodes que nous avons étudiées sont des problèmes de régulation de trafic ferroviaire et d'ordonnancement d'atelier.

5-2-1- *Problèmes des trains au départ*

Cette série de problèmes nous a été communiquée par M. GENETE de la SNCF le 7.2.1974.

20 voies de chemin de fer convergent vers une voie unique. Sur chacune de ces 20 voies peut se trouver au plus un train en attente pour pénétrer sur la voie unique. Cette voie est segmentée en six cantons ; le tableau A de l'annexe 1 nous fournit le temps d'occupation, en unité de temps, de chacun des 6 cantons pour chacun des 20 trains. Remarquons sur ce tableau que les trains ont en général des vitesses différentes, c'est-à-dire que les temps d'occupation des cantons sont différents selon les trains. La contrainte à res-

pecter est la suivante : il doit toujours exister au minimum un canton libre entre deux trains consécutifs. On cherchera à minimiser le temps entre l'entrée du premier train sur le premier canton et la sortie du dernier train du dernier canton. Les trains seront considérés avec une longueur nulle.



Remarque

En reprenant les notations introduites au paragraphe 1-2, ce problème peut se définir sous la forme $n/6/G_2/(PM1)/F_{MAX}$:

- n étant le nombre de pièces à produire (de trains à faire circuler), nous avons pris $7 \leq n \leq 20$.

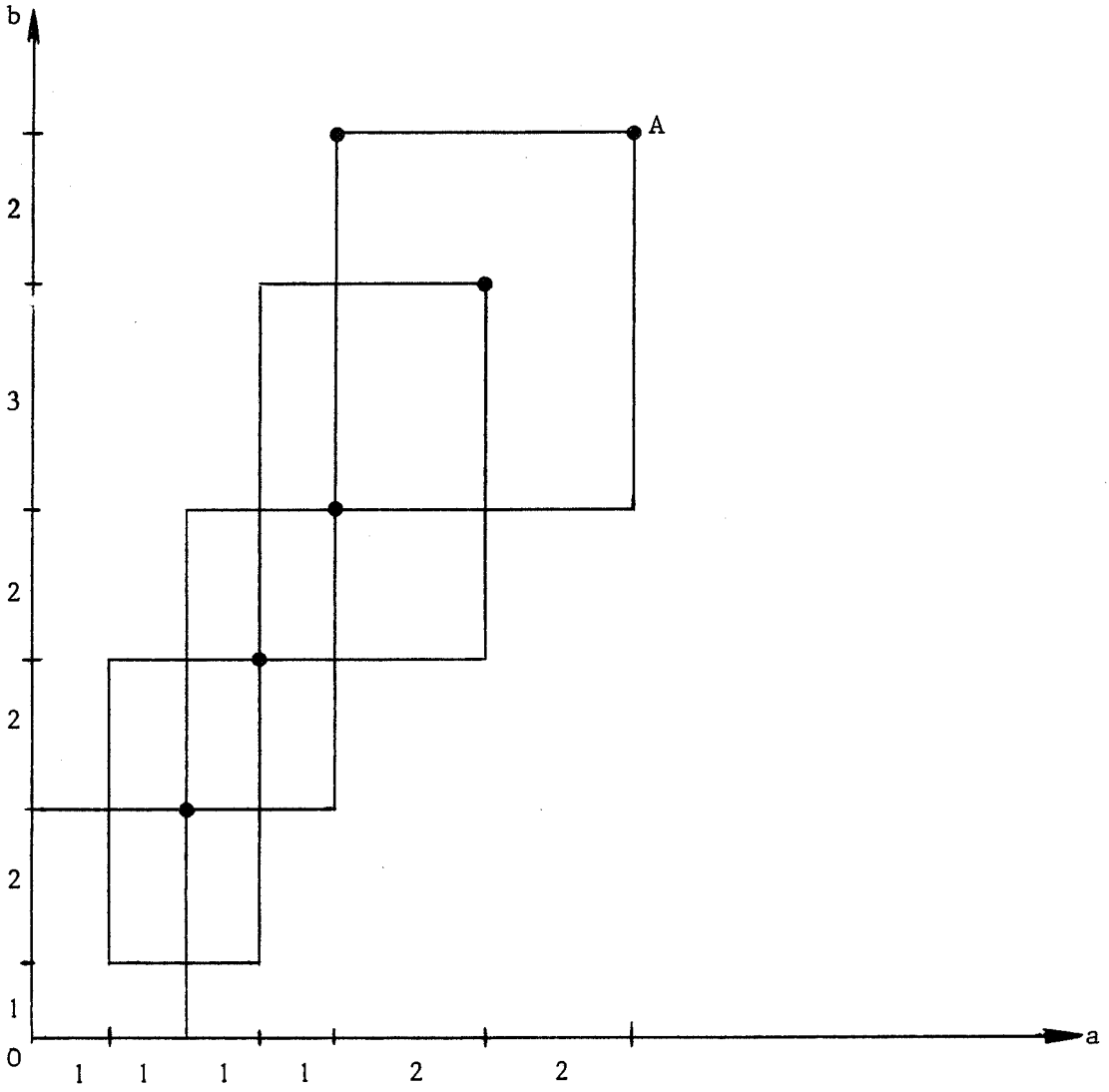
- 6 étant le nombre de machines, (ici de cantons).

- G_2 signifie que les gammes sont identiques, (tous les trains circulent dans le même sens).

- PM1 signifie qu'il existe des durées de réemploi, (un canton libre au minimum entre chaque train).

- F_{MAX} signifie que le critère à minimiser est le temps entre l'arrivée de la première pièce sur la première machine (du premier train dans le premier canton) et la fin du passage sur la dernière machine (du dernier train dans le dernier canton).

Ce problème particulier peut se formaliser comme celui du voyageur de commerce. La distance entre deux villes étant remplacée par le temps séparant l'arrivée de deux trains consécutifs. Nous avons cependant cherché sa solution sans tenir compte de cette représentation.



Le traitement d'un tel problème par calculateur spécialisé conduit à représenter, dans chacun des plans, les contraintes de succession des deux trains concernés. Le schéma ci-dessus représente le plan d'analyse des trains a et b (voir tableau A de l'annexe 1).

Le nombre N de variables des problèmes de la batterie varie de 7 à 20. Chacun de ces problèmes est construit en choisissant N parmi les 20 trains (a, b, ..., u) des tableaux B et C de l'annexe 1.

5-2-2- *Problèmes d'ordonnancement d'atelier*

Deux séries de problèmes ont été traitées. Elles ont été extraites d'exemples fournis par MUTH et THOMPSON [13]. Ces données figurent en annexe 2.

Remarque :

On pourrait aussi imaginer traiter des problèmes de programmation linéaire [16] par les calculateurs spécialisés. Il s'agirait alors de trouver les valeurs des variables X_1, X_2, ..., X_n qui maximisent la relation :

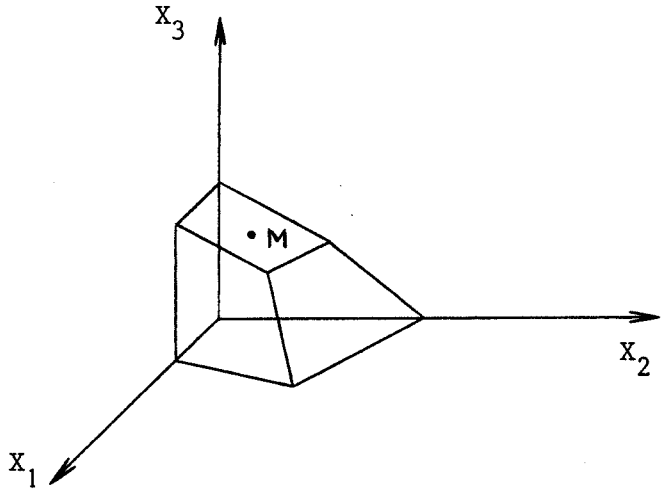
$$F = C_1X_1 + C_2X_2 + \ldots + C_nX_n \qquad (I)$$

et vérifient les conditions suivantes :

$$\left.\begin{array}{l} a_{11}X_1 + a_{12}X_2 + \ldots + a_{1n}X_n \leqslant b_1 \\ a_{21}X_1 + a_{22}X_2 + \ldots + a_{2n}X_n \leqslant b_2 \\ \cdot \qquad\qquad\qquad\qquad \cdot \\ \cdot \qquad\qquad\qquad\qquad \cdot \\ \cdot \qquad\qquad\qquad\qquad \cdot \\ a_{m1}X_1 + a_{m2}X_2 + \ldots + a_{mn}X_n \leqslant b_m \\ X_1, X_2, \ldots X_n \text{ positives ou nulles} \end{array}\right\} \quad (II)$$

Les C_k et les a_{ij} sont des données.

Prenons un exemple avec n = 3. Les conditions (II) signifient que les valeurs cherchées des variables X_1, X_2, X_3 sont les coordonnées d'un point M intérieur au polyèdre représenté sur la figure suivante :



La relation (I) signifie que X_1, X_2, X_3 sont les coordonnées de M appartenant à un plan dont on connaît la direction. M est donc l'intersection d'un polyèdre et d'un plan de direction donnée. L'extérieur du polyèdre constitue les contraintes du problème. Malheureusement, contrairement aux problèmes d'ordonnancement, ces contraintes n'ont pas leurs arêtes parallèles aux axes de coordonnées et les projections planes n'ont plus de significations, il n'a donc pas été possible de traiter simplement des problèmes de programmation linéaire par les calculateurs spécialisés.

5 - 3. RESULTATS ET COMMENTAIRES

5-3-1- *Problèmes traités sur l'optimateur CYBCO*

Nous avons traité certains exemples de trains au départ figurant en annexe 1, tableaux B et C. Nous avons rassemblé, page suivante, les différents résultats obtenus, on trouve :

- colonne 1 : référence du jeu d'essai, le premier chiffre indiquant le nombre de variables considérées.

- colonne 2 : tempsTde traitement par le système formé de l'optimateur et de l'ordinateur Télémécanique T1600.

- colonne 3 : résultat X de l'ordonnancement (en unités de temps ou nombre de pas) obtenu par le système CYBCO.

- colonne 4 : espérance mathématique $\overline{X}$ des solutions égale à la moyenne de tous les ordonnancements possibles.

- colonne 5 : coût de la meilleure solution obtenue X^* grâce à un programme de recherche en arbre mis au point par B. JULLIEN [14] et testé sur l'ordinateur Philips P1175.

- colonne 6 : temps mis par ce programme de recherche en arbre.

- colonne 7 : pouvoir optimisant de l'optimateur selon la définition de M. GENETE,

$$P = \frac{X - \overline{X}}{X^* - \overline{X}}$$

N°ex	durée	résultat d'ordon;	espérance	meilleure solution	temps CPU P1175	P;O.
701	32"	169	182	138	2"	20,5%
702	28"	179	202	160	2"	54,7%
703	40"	162	185	157	2"	82,1%
704	21"	109	132	100	2"	71,9%
705	28"	205	208	175	2"	9,1%
706	29"	184	211	176	2"	77,1%
801	47"	228	229	182	2,5"	2,1%
802	44"	165	174	134	2,5"	22,5%
803	56"	176	202	150	2,5"	50,0%
804	49"	160	196	150	2,5"	78,3%
805	41"	203	226	173	2,5"	43,4%
806	27"	106	136	106	___	100,0%
901	1'31"	252	273	211	3"	33,9%
902	53"	164	171	134	3"	18,9%
903	1'25"	184	223	171	3"	75,0%
904	49"	169	185	153	3"	50,0%
905	1'17"	199	210	165	3"	24,4%
906	1'13"	248	257	211	3"	19,6%
1001	1'43"	232	241	179	3"	14,5%
1002	2'09"	266	288	219	3"	31,9%
1003	1'42	244	269	216	3"	47,2%
1004	1'19"	178	208	165	3"	69,8%
1005	2'09"	262	298	234	3"	56,3%
1006	2'08"	267	301	227	3"	45,9%
1101	2'20"	237	250	193	4"	22,8%
1102	2'58"	256	294	234	4"	63,3%
1103	2'15"	242	271	203	4"	42,6%
1104	3'00"	323	348	266	4"	30,5%
1105	2'56"	196	243	186	4"	82,5%
1106	3'33"	255	295	226	4"	58,0%
2001	19'34"	450	504	364	12"	38,6%

Pouvoir optimisant moyen : 0,47

Pour évaluer le temps de calcul T, considérons le temps d'exécution des différentes parties de la méthode décrite en 3-4-1:

1 - scrutation des $\frac{N(N-1)}{2}$ plans de projection, temps proportionnel à N(N-1),

2 - tri des N processus selon l'ordre décroissant des valeurs C(I), I variant de 1 à N, temps proportionnel à N. Log N,

3 - pour chaque processus T_I qui progresse, mise à jour du tableau des incompatibilités, temps proportionnel à N(N-1),

4 - les 3 points précédents doivent être répétés à chaque pas de la trajectoire, donc temps proportionnel à C, longueur de la trajectoire.

λ, α', β', γ' étant des coefficients de proportionnalité, nous avons la relation :

$$T = \lambda C\left[\alpha' \frac{N(N-1)}{2} + \beta' N.\text{Log } N + \gamma' N(N-1)\right]$$

Ce qui s'écrit :

$$T = \alpha C\left[N^2 + \beta N + \gamma N.\text{ Log } N\right]$$

Il a été possible d'évaluer approximativement le temps propre à l'optimateur grâce à un programme test effectuant seulement des entrées-sorties avec l'optimateur ou des traitements de contraintes. On a constaté qu'une entrée ou une sortie coûtait environ 10 µs et le traitement d'une contrainte environ 5µs. Nous avons vu, au paragraphe 3-3 qu'un cycle d'initialisation, un cycle résultat ou le traitement d'une contrainte nécessitaient chacun deux échanges optimateur-ordinateur.

Dans les exemples traités, il y a 5 contraintes par plan et dans le test 1001 où N = 10, le temps T_o propre à l'optimateur est $T_o = \left[\left[40 + 5(5+20)\right] . \frac{10.9}{2}\right].232 = 1{,}72 \times 10^6$ µs ≃ 2 secondes, ce qui est très faible par rapport au temps total T = 1 mn 43 s.

5-3-2- *Simulation des calculateurs spécialisés*

Nous avons traité par simulation essentiellement deux types de problèmes : les trains au départ et l'ordonnancement d'atelier.

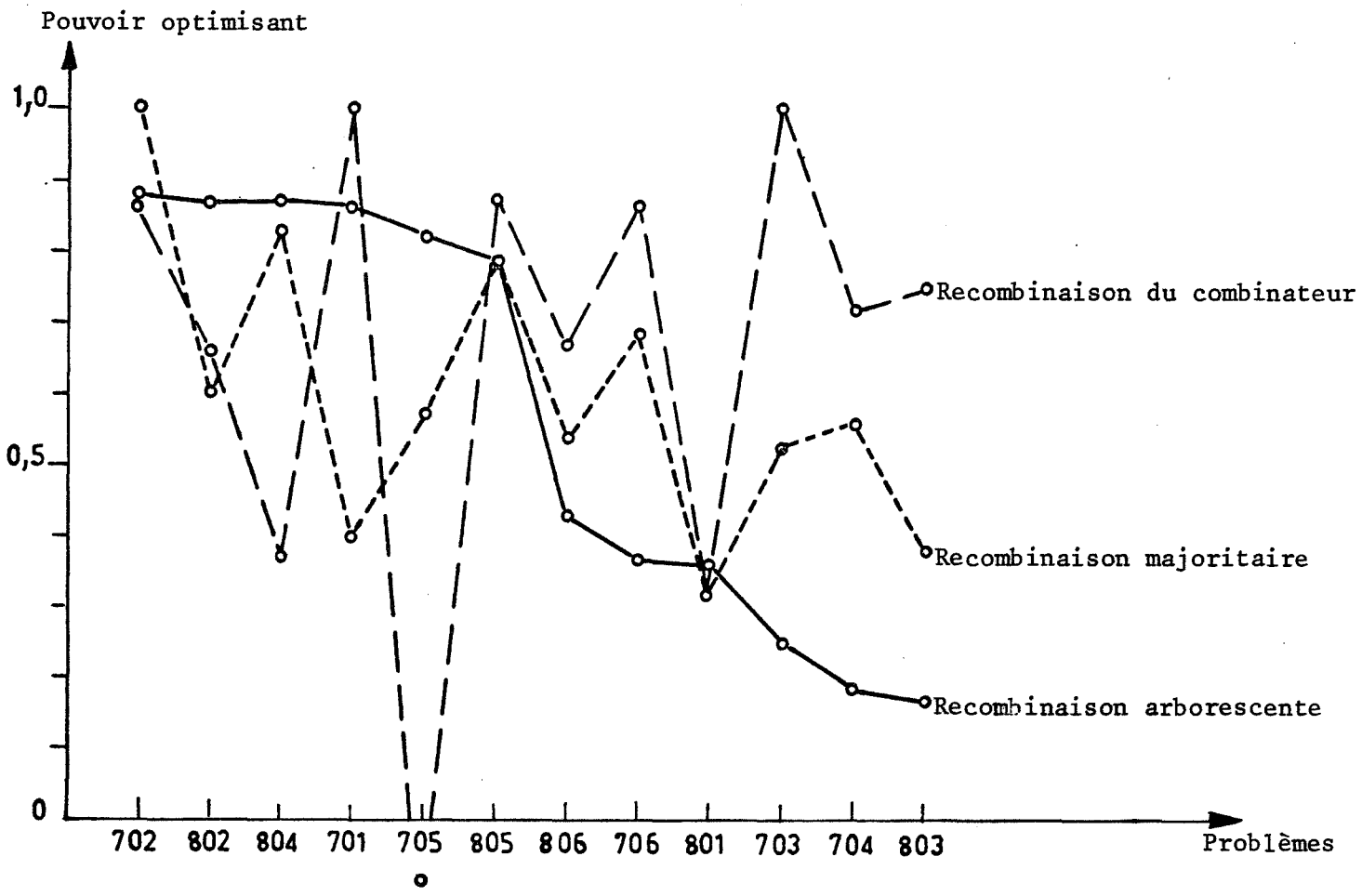
. Trains au départ

- Résultats.

Méthodes / N° des exercices	Algorithme B & B		Recombinaison arborescente		Recombinaison majoritaire		Recombinaison du combinateur		Espérance
	Temps	Solution	Temps	Solution	Temps	Solution	Temps	Solution	
701	2 "	138	229 "	144	61 "	164	77 "	138	182
702	2 "	160	278 "	165	67 "	160	87 "	164	202
703	2 "	157	274 "	178	64 "	170	85 "	157	185
704	2 "	100	178 "	126	40 "	114	58 "	109	132
705	2 "	175	312 "	181	77 "	189	127 "	213	208
706	2 "	176	304 "	198	77 "	187	112 "	181	211
801	2,5 "	182	526 "	212	103 "	214	179 "	214	229
802	2,5 "	134	346 "	139	62 "	150	88 "	148	174
803	2,5 "	150	447 "	193	77 "	182	106 "	163	202
804	2,5 "	150	407 "	156	74 "	158	139 "	179	196
805	2,5 "	173	477 "	184	89 "	184	136 "	180	226
806	2,5 "	106	298 "	123	48 "	112	80 "	116	136

- Pouvoirs optimisants.

Méthodes / Exercices	Recombinaison arborescente	Recombinaison majoritaire	Recombinaison du combinateur
701	0,86	0,40	1
702	0,88	1	0,86
703	0,25	0,53	1
704	0,19	0,56	0,72
705	0,82	0,57	- 0,15
706	0,37	0,68	0,86
801	0,36	0,32	0,32
802	0,87	0,60	0,65
803	0,17	0,38	0,75
804	0,87	0,83	0,37
805	0,79	0,79	0,87
806	0,43	0,54	0,67



Nous avons représenté ici le pouvoir optimisant des différentes méthodes en fonction des exercices traités. Ces exercices figurent en abscisse selon les valeurs décroissantes du pouvoir optimisant de la recombinaison arborescente.

Nous constatons que cette recombinaison arborescente ne donne pas des résultats systématiquement meilleurs qu'une méthode rudimentaire telle que la recombinaison majoritaire. Or comme la recombinaison arborescente est optimale vis-à-vis de l'hypothèse énoncée en 3-4-2 et disant que la trajectoire spatiale n'est que légèrement plus longue que la plus longue des trajectoires planes, cela prouve que cette hypothèse est inexacte.

. Ordonnancement d'atelier

En plus des résultats fournis par les recombinaisons, nous avons figuré ici les résultats obtenus par les méthodes logicielles que nous avons définies précédemment (C.D., L.D., L.T.R. et apprentissage). Dans la colonne exercice, le première nombre représente le nombre de tâches, le second le nombre de machines.

1ère série d'exercices

- Résultats.

Exemples	Recombinaison arborescente		Recombinaison majoritaire		Recombinaison ducombinateur		C.D.		L.D.		L.T.R.		Apprentissage			$\overline{X}$	X*
	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U1	Temps C P U2	Résultat		
5 x 5	38"	57	16"	62	20"	63	0,42"	54	0,42"	62	0,42"	66	2,5"	141"	51	75	45
6 x 5	72"	63	23"	61	40"	83	0,55"	60	0,55"	65	0,55"	66	14"	194"	52	83	48
7 x 5	117"	61	31"	66	47"	88	0,71"	72	0,71"	72	0,71"	72	12"	252"	58	92	53
8 x 5	204"	65	40"	73	68"	105	0,85"	79	0,85"	80	0,85"	76	23"	300"	62	98	57
9 x 5	358"	76	54"	76	74"	82	0,98"	79	0,98"	81	0,98"	77	276"	695"	64	108	60
10 x 5	631"	93	74"	86	109"	110	1,20"	87	1,20"	93	1,20"	89	290"	848"	75	123	67

NOTE :

Pour l'apprentissage, le temps CPU1 est le temps nécessaire, avec l'ordinateur Philips P1175, pour atteindre le meilleur ordonnancement fourni par cette méthode, le temps CPU2 est le temps nécessaire à la convergence de l'apprentissage.

- Pouvoirs optimisants

Méthodes / Exerc.	Recomb. arbor.	Recomb. majorit.	Recomb. du comb.	C.D.	L.D.	L.T.R.	Apprentissage
5 x 5	0,60	0,43	0,40	0,70	0,43	0,30	0,80
6 x 5	0,57	0,63	0	0,66	0,51	0,48	0,88
7 x 5	0,79	0,67	0,10	0,51	0,51	0,51	0,87
8 x 5	0,80	0,61	-0,17	0,46	0,44	0,53	0,87
9 x 5	0,67	0,67	0,54	0,60	0,56	0,64	0,91
10 x 5	0,53	0,66	0,23	0,64	0,53	0,60	0,86

Pour déterminer les pouvoirs optimisants de ces méthodes dans le cas des problèmes d'atelier, nous reprenons la formule $P = \frac{\overline{X}-X}{\overline{X}-X^*}$.
Explicitons dans ce cas l'obtention de $\overline{X}$ et X^*.

Obtention de $\overline{X}$:

Un ordonnancement quelconque est défini par l'ordre de passage des pièces sur les différentes machines. Pour un problème à n pièces et m machines, il y a donc $(n!)^m$ ordonnancements possibles. $\overline{X}$ est l'espérance mathématique calculée à partir de tirages aléatoires dans l'ensemble des ordonnancements possibles.

Obtention de X^*:

X^* est inférieur ou égal à OP, ordonnancement optimal inconnu. De plus X^* doit vérifier les deux inégalités suivantes :

. $X^* \geq$ MAXP

MAXP représente le temps d'usinage maximal des différentes pièces. Pour le problème 5 x 5 par exemple, MAXP = 33, obtenu pour la pièce 2 (voir les données en annexe 2).

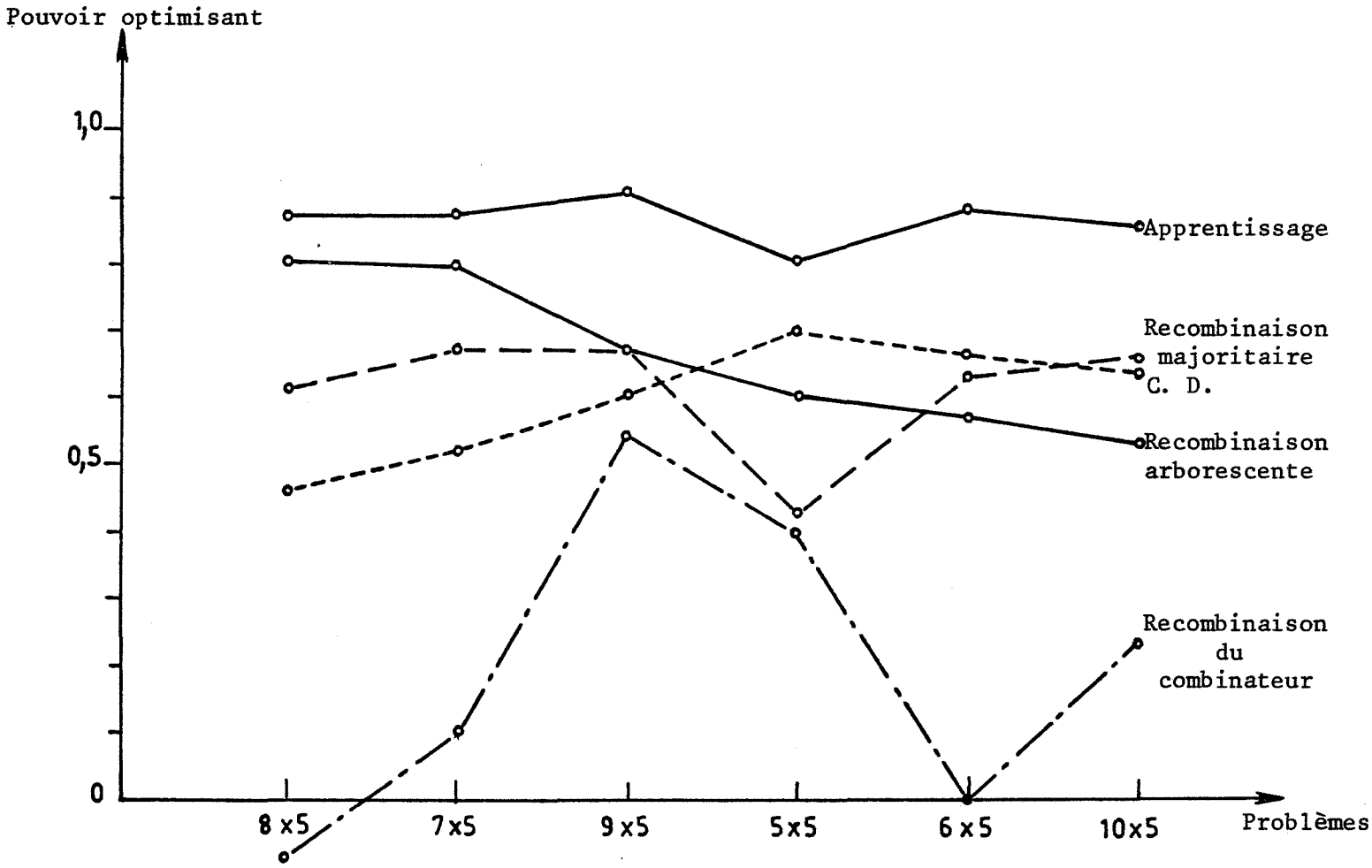
. $X^* \geq$ MAXM

Calculons, pour chaque machine, la somme des temps d'exécution des pièces qui leur sont affectées, soit M(I) ce temps pour la machine I. A ce temps, ajoutons d'une part le délai minimal d'alimentation A(I), (nul s'il existe une pièce dont la gamme commence par la machine I) et d'autre part, le temps minimal de terminaison F(I), (nul s'il existe une pièce dont la gamme se termine par la machine I). MAXM est le maximum de ces temps relatifs à l'ensemble des machines :

$$MAXM = \sup_{I} \left[M(I) + A(I) + F(I) \right]$$

Pour le problème 5 x 5, MAXM = 45, obtenu pour la machine 2 avec M(2) = 31, A(5) = 0 et F(5) = 14.

L'ordonnancement optimal du problème est supérieur ou égal à la plus grande des 2 valeurs MAXP et MAXM, c'est pourquoi nous prendrons : $X^* = \sup \left[MAXP, MAXM \right]$, soit 45 pour le problème 5 x 5.



2ème série d'exercices

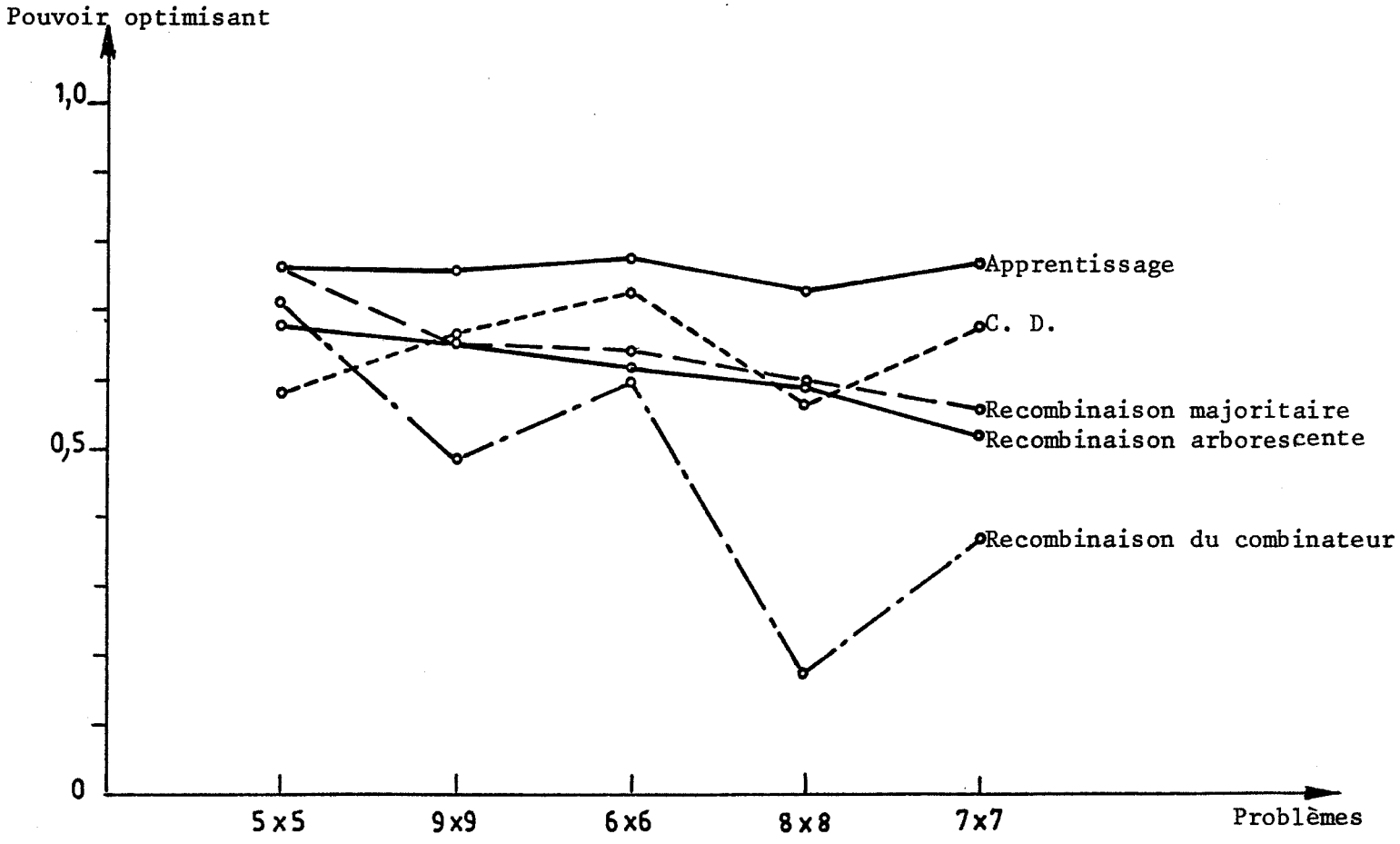
Ces exercices ont été obtenus en considérant pour le problème 6 x 6, les 6 premières pièces et les 6 premières machines du tableau B de l'annexe 2, pour le problème 7 x 7, les 7 premières pièces et les 7 premières machines, et ainsi de suite.

- Résultats :

Exemples.	Recombinaison arborescente		Recombinaison majoritaire		Recombinaison du combinateur		C.D.		L.D.		L.T.R.		Apprentissage			$\overline{X}$	X*
	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U	Résultat	Temps C P U1	Temps C P U2	Résultat		
5 x 5	42 "	59	19 "	56	22 "	58	0,4 "	63	0,4 "	60	0,4 "	62	2 "	128 "	56	85	47
6 x 6	92 "	72	33 "	71	40 "	73	0,76"	67	0,76"	68	0,76"	73	38 "	318 "	65	100	55
7 x 7	174 "	89	55 "	86	73 "	98	1,21"	85	1,21"	95	1,21"	92	6 "	400 "	73	121	59
8 x 8	330 "	100	92 "	99	116 "	134	1,87"	102	1,87"	95	1,87"	123	352 "	1223 "	88	149	66
9 x 9	577 "	113	150 "	114	186 "	130	3 "	112	3 "	123	3 "	141	762 "	2085 "	103	178	80

- Pouvoirs optimisants :

Méthodes / Exempl.	Recomb. arbores.	Recomb. major.	Recomb. du comb.	C.D.	L.D.	L.T.R.	Apprentis.
5 x 5	0,68	0,76	0,71	0,58	0,66	0,60	0,76
6 x 6	0,62	0,64	0,60	0,73	0,71	0,60	0,78
7 x 7	0,52	0,56	0,37	0,68	0,42	0,47	0,77
8 x 8	0,59	0,60	0,18	0,57	0,65	0,31	0,73
9 x 9	0,66	0,65	0,49	0,67	0,56	0,38	0,76



A la lecture de ces résultats, on peut faire les commentaires suivants :

1 - Qualité de la solution

Pour les problèmes de trains au départ, les calculateurs spécialisés n'ont pratiquement jamais conduit à des ordonnancements plus performants que ceux obtenus par la méthode heuristique appropriée.

Pour les problèmes d'atelier, les résultats obtenus par les calculateurs spécialisés sont même moins bons, en général, que ceux obtenus par les diverses heuristiques. Le combinateur, qui donnait des résultats aussi valables que les autres recombinaisons pour des problèmes à contraintes diagonales emboîtées (voir début du paragraphe 5-3-2), donne des résultats significativement moins bons lorsque les contraintes sont disjointes.

Cependant, dans les sociétés qui mettent au point le combinateur et l'optimateur, certains responsables pensent qu'il existe des algorithmes de recombinaison non encore découverts, et dont la mise au point révolutionnera le pouvoir optimisant en offrant à ces calculateurs spécialisés des débouchés universels. Nous nous garderons bien d'émettre un jugement aussi affirmatif à cet égard !

2 - Temps de calcul

Il apparaît que le temps de la recombinaison arborescente (paragraphe 3-4-2) est nettement plus élevé que celui des deux autres recombinaisons (paragraphes 3-4-1 et 4-2). On constate que les temps de calcul des recombinaisons sont proportionnels au nombre de plans scrutés et au résultat de l'ordonnancement.

Il est difficile de comparer les temps de calcul des recombinaisons et ceux des méthodes logicielles. En effet, les temps des recombinaisons pourront être grandement améliorés grâce au câblage de diverses procédures alors que ceux des méthodes logicielles sont pratiquement incompressibles. De plus, pour les problèmes d'atelier comme pour ceux des trains au départ, ce sera le temps nécessaire à l'obtention de la première décision (du premier pas) qui sera immédiatement utile, les autres éléments de décision pouvant intervenir plus tard.

Expression du temps de calcul des recombinaisons

. Recombinaison majoritaire

Nous avons vu, en 5-3-1, que le temps de calcul T, en secondes, d'un ordonnancement à N variables, de C pas, était :

$$\boxed{T = \alpha C \left[N^2 + \beta N + \gamma N \text{ Log } N\right]}$$

. Recombinaison arborescente

Les différentes étapes de cette méthode (voir 3-4-2) sont les suivantes :

1 - Scrutation des $\frac{N(N-1)}{2}$ plans de projection, temps proportionnel à N(N-1),

2 - Tri des 2N(N-1) distances obtenues par le calculateur spécialisé, temps proportionnel à N(N-1)LogN(N-1), soit à N(N-1)LogN en confondant Log N et Log(N-1),

3 - Scrutation d'un arbre binaire tronqué pour chaque processus, temps proportionnel à $N\ K^N$ (avec $1 < K < 2$),

4 - Répétition des 3 étapes précédentes à chaque pas de la trajectoire, temps proportionnel à C longueur de la trajectoire.

Nous obtenons finalement la relation :

$$\boxed{T = \alpha C\left[N\ K^N + \beta N^2 + \gamma N + \delta N^2 \text{Log } N + \lambda N \text{ Log } N\right]}$$

. Recombinaison du combinateur

Chaque étape de la méthode exposée en 4-2 est en N^2 ou N sauf la dernière (définition du pas) qui est en 2^N, ce qui conduit à la relation :

$$\boxed{T = \alpha C\left[2^N + \beta N^2 + \gamma N\right]}$$

3 - Taille mémoire

Remarquons enfin que la taille mémoire nécessitée par la simulation de chaque recombinaison est de l'ordre de grandeur de celle nécessitée par la programmation des méthodes purement logicielles.

CONCLUSION

L'étude que nous avons menée nous a permis de cerner les caractéristiques des calculateurs spécialisés imaginés à la suite des travaux du Docteur SAUVAN et expérimentés aujourd'hui par diverses sociétés telles que CYBCO et la SNCF.

Nous constatons que ces machines fournissent toujours un bon résultat en un temps très bref pour tous les problèmes plans, même complexes, car alors seule l'analyse bidimensionnelle, algorithme efficace, est utilisée. Par contre, lorsqu'on aborde les problèmes très combinatoires à plus de deux processus, il en va tout autrement car alors il faut utiliser une heuristique de recombinaison et aucune de celles-ci jusqu'à maintenant ne nous a fourni des résultats vraiment satisfaisants.

En effet, la mise en oeuvre des recombinaisons influe sur :

- la qualité de la solution obtenue,
- le temps d'obtention de la solution,
- le coût d'exploitation du système informatique.

1. QUALITE DE LA SOLUTION OBTENUE

Nous avons dû constater l'absence actuelle de recombinaisons vraiment efficaces. Celles que nous avons utilisées s'avèrent plus ou moins performantes selon la nature du problème traité. Jusqu'à maintenant, toutes les recombinaisons ne sont que des heuristiques approchées car il n'est pas possible en général de construire une trajectoire spatiale optimale à partir de trajectoires planes optimales.

Les recombinaisons que nous avons étudiées ne fournissent que des solutions sous-optimales qui n'ont jamais été systématiquement meilleures que celles obtenues par les méthodes purement logicielles assez élémentaires prises comme références.

2. TEMPS D'OBTENTION DE LA SOLUTION

Comme la recombinaison scrute tous les plans de projection et construit la trajectoire pas par pas, le temps de calcul est toujours au moins proportionnel au carré du nombre de variables et à la longueur de la trajectoire, ce qui n'est pas vrai pour les approches logicielles. Les temps de calcul des recombinaisons simulées sur ordinateur Philips P1175 sont sensiblement supérieurs à ceux obtenus pour les méthodes logicielles programmées sur le même ordinateur. Ces temps sembleraient pouvoir être améliorés grâce à de meilleures recombinaisons, mais certainement pas suffisamment. Pour diminuer très sensiblement ce temps, il faudrait câbler tout ou partie de la recombinaison mais ceci s'avère difficile à réaliser aujourd'hui étant donné qu'il n'existe pas de recombinaison vraiment performante capable de résoudre différents types de problèmes d'ordonnancement. Certains chercheurs envisagent actuellement des recombinaisons spécifiques pour chaque type de problèmes, ce qui compliquerait évidemment le câblage.

3. COUT D'EXPLOITATION DU SYSTEME MIS EN OEUVRE

Les programmes de recombinaison et de pilotage des calculateurs spécialisés nécessitent le couplage de ces derniers à un ordinateur et utilisent une place mémoire du même ordre de grandeur que les programmes purement logiciels que nous avons employés. Remarquons cependant, que le câblage de la recombinaison réduirait la taille de la mémoire utilisée qui resterait néanmoins encore importante. A l'heure actuelle, le coût de résolution d'un problème d'ordonnancement par une méthode logicielle est inférieur au coût de résolution par un calculateur spécialisé associé à un ordinateur car dans le premier cas on fait l'économie d'une machine bidimensionnelle.

Ainsi, il semble que les recherches concernant les calculateurs spécialisés marquent le pas et ne donnent pas suite à tous les espoirs mis en elles il y a quelques années lorsqu'on avait découvert cette nouvelle et originale manière d'aborder les problèmes d'ordonnancement. La cause principale des difficultés rencontrées est le manque de recombinaisons efficaces : elles ne sont que de simples heuristiques ayant très peu de justifications théoriques. En effet, toutes les heuristiques de recombinaison que nous avons étudiées sont basées sur l'hypothèse suivante : plus le nombre de pas de la plus longue des solutions planes utilisées est faible, plus la trajectoire réelle dans l'espace de dimension N

est courte. C'est cette hypothèse que nous appliquons dans la recombinaison arborescente puisque l'heuristique utilisée consiste directement à éliminer les solutions planes les plus longues. Quant à la recombinaison majoritaire, elle favorise dans chaque plan les solutions les plus courtes. La recombinaison du combinateur intègre simultanément ces deux variantes. Malheureusement cette hypothèse fondamentale ne peut pas être vérifiée en général, ce qui conduit parfois à des solutions nettement sous-optimales.

On constate que les chercheurs qui ont conçu les calculateurs spécialisés n'ont pas suffisamment respecté les règles à observer lors de la résolution d'un problème complexe par décomposition en sous-problèmes plus simples :

1. Prouver que la solution globale du problème ne sera pas altérée si elle est obtenue à partir de solutions de problèmes partiels.

2. Résoudre séparément les différentes parties du problème.

3. Recombiner les différentes solutions obtenues pour avoir la solution globale.

Les recherches entreprises jusqu'à maintenant se sont surtout concentrées sur les points 2 et 3 en négligeant le point 1 qui aurait dû être abordé avant les autres.

Au lieu de chercher à mettre au point de nouveaux calculateurs spécialisés pour résoudre les problèmes d'ordonnancement selon des méthodes inspirées des travaux du Docteur SAUVAN, il apparaît plus intéressant de mettre en matériel des logiciels qui ont prouvé être intrinséquement déjà extrêmement rapides. On peut espérer de cette façon que le matériel reprenant ce logiciel soit encore plus rapide. C'est ainsi que la thèse de B. JULLIEN [14] a montré l'intérêt d'un algorithme qui, en partant d'une bonne solution d'un problème d'ordonnancement obtenue par une heuristique quelconque, améliore cette solution en cherchant toutes les interversions de tâches qui permettent d'améliorer le critère. Une machine qui générerait des permutations de tâches à partir d'une solution de départ donnée, en réalisant des interversions de rang 1 (permutation de 2 tâches), de rang 2 (permutation de 3 tâches) ou de rang 3 (permutation de 4 tâches) en respectant

les contraintes du problème, pourrait alors être efficace. Une permutation étant générée, il faudrait ensuite fournir l'ordonnancement des tâches à un autre calculateur, soit général, soit spécialisé, qui donnerait la valeur du critère en tenant compte de la permutation. Si cette permutation améliore le critère, l'ordonnancement obtenu devient alors une nouvelle solution de départ pour une reprise de l'algorithme. Il serait certainement possible d'orienter ces permutations par un apprentissage, en favorisant certaines formes de permutations qui ont déjà fourni de bons résultats. Une autre approche, dont le câblage pourrait aussi être étudié, serait l'approche booléenne [15], il est très possible qu'elle puisse conduire à des résultats très intéressants.

Cependant, il s'avère que dans beaucoup de cas, le problème de la saisie de données est plus important que celui de trouver l'ordonnancement optimal. On peut donc se demander si, plutôt que de mettre au point des ordinateurs spécialisés de plus en plus rapides, on ne pourrait pas, pour résoudre les problèmes d'ordonnancement, mettre l'accent sur les communications homme-machine aptes à une meilleure saisie de données, et que peut-être, par un traitement fort simple de ces données, on peut déjà obtenir de l'humain qui est à l'autre bout de la ligne, une bonne solution.

L'ordonnancement jouera certainement à l'avenir un rôle de plus en plus grand, mais il n'est pas pensable d'avoir des machines spécialisées qui résolvent les problèmes d'ordonnancement vis-à-vis de n'importe quelles hypothèses, il faut donc fixer les hypothèses (voir le paragraphe 1-1-1) avant de concevoir d'abord l'algorithme, et ensuite la machine spécialisée pour résoudre le problème. D'autre part, il n'est pas question de faire de bonnes machines matérielles avant d'avoir de bons algorithmes. Cette thèse nous aura appris qu'il faut être extrêmement prudent sur les algorithmes dès le départ, il faut se méfier des à-priori tel celui qu'une simulation du cerveau humain peut résoudre n'importe quel type de problèmes, ou tel celui qu'un ordinateur analogique aurait des possibilités de parallèlisme bien supérieures à celles d'un ordinateur digital et qu'on pourrait donc résoudre des problèmes forts complexes en simulant des comportements analogiques, ces deux à-priori se sont révélés être tout à fait faux. Ce genre d'hypothèses ne tient donc pas et les algorithmes doivent être testés à fond avant d'être implémentés sous forme matérielle.

A N N E X E S

ANNEXE 1

cantons / trains	1	2	3	4	5	6
a	1	1	1	1	2	2
b	1	2	2	2	3	2
c	1	1	1	2	4	3
d	1	4	3	3	2	2
e	1	4	3	3	4	3
f	1	3	3	3	3	3
g	1	4	4	2	2	4
h	1	1	2	3	4	3
j	1	10	22	11	23	4
k	1	13	15	11	18	23
l	1	25	4	24	6	10
m	1	4	25	19	21	18
n	1	10	4	6	11	4
o	1	6	8	6	4	4
p	1	7	15	19	6	8
q	1	10	11	6	4	8
r	1	6	2	2	5	8
s	1	15	3	11	8	4
t	1	6	12	25	13	5
u	1	7	10	11	12	6

TABLEAU A - TEMPS D'OCCUPATION DES CANTONS PAR LES DIFFERENTS TRAINS.

Numéro des exercices		TRAINS CONSIDERES
3	01	g s u
	02	h m u
	03	g k q
	04	b g j
	05	b e s
	06	h j r
4	01	h k m o
	02	c j m u
	03	f l o p
	04	e g j r
	05	b m p q
	06	c e n q
5	01	a c k r s
	02	c e g h s
	03	b e j m t
	04	e j m p q
	05	b h o p t
	06	a d f m r
6	01	a k m n p r
	02	c g n p q t
	03	a h k n r u
	04	e g l m o q
	05	c e h n o r
	06	a d l m t u
7	01	a b f k n p u
	02	d e f k m q r
	03	h j n p r t u
	04	b c g o p r u
	05	d e m n q t u
	06	e j m o q r s
8	01	e g j m n o q u
	02	a e f g h k n p
	03	b d f h l m r s
	04	a e f k p q r s
	05	c f g h k m s u
	06	b c d e o p r s
9	01	c d j k l p q r s
	02	a b e f g p q s u
	03	a e f h k m n o r
	04	b d e g h m n o s
	05	b e f g m n p q r
	06	e f g j l o p q t
10	01	a b d e f l m n q s
	02	b c j l m n o q r u
	03	b c f g j m n o s t
	04	a c d g h n p q t u
	05	f g h j l m n p r u
	06	d e h k l m n p r u
11	01	a b c d e g m n o p t
	02	c e f g h l m p r t u
	03	b c d h l m n q r s u
	04	c f g j k l m o p s u
	05	e f h l n o p q r s u
	06	a c h j m n o q r t u

TABLEAU B

Numéro des Exercices		TRAINS CONSIDERES
12	01	a b d e f k l m o q r u
	02	b c e j l m n o r s t u
	03	b c g h j l m n o p s t
	04	a d e g j k l m o q s t
	05	f g j k l m n o q r s t
	06	a f g k l m o p r s t u
13	01	c d e f g h j l m n o s u
	02	a c f g h j k m n o p t u
	03	a b c g h j k l o r s t u
	04	a b c f g h l m n r s t u
	05	a b c d e f h l m o p q s
	06	a b d e g h l m n o t u
14	01	a b c e f g k l o p q r t u
	02	c e g h j k l m o q r s t u
	03	a d g h k l m n o p r s t u
	04	b c d e g j l m n o p q r s
	05	c d e f g h j k l n r s t u
	06	c d e f h k m n o p q s t u
15	01	c d e f g h j k l m n p q r s
	02	b c f g h j k l m o p q s t u
	03	b c d e f g h j k m n p q r s
	04	c d e f g h j k m n p q r s u
	05	a b f g j k l m n p q r s t u
	06	c d e h j k l m n o p q s t u
16	01	b e f g h j k l m n o p r s t u
	02	a b c d f g h k l m o p q r s t
	03	a b c d e g j k l n o p q r s t
	04	a b c d e f g h j k m n o r s t
	05	a b d e h j k l m n o p r s t u
	06	a b c f g h j l m n p q r s t u
17	01	a b c d e f h k l n o p q r s t u
	02	a c e f g h j k l m n o p r s t u
	03	a b c d e f g h k m n p q r s t u
	04	a b c e f g h j k l m o p q s t u
	05	b c d e f g j l m n o p q r s t u
	06	a b c d e f g h k l m n p q r s u
18	01	a c d e f g h j k l m n o p q r s u
	02	a c d e f g h j k m n o p q r s t u
	03	a b c d e f g h k l n o p q r s t u
	04	a b c d e f h j l m n o p q r s t u
	05	a b c d e f g h j l m o p q r s t u
	06	a c d e f g h j k l m n o p q r s t
19	01	a b c d e f g h j k l m n o p q r s t
	02	b c d e f g h j k l m n o p q r s t u
	03	a c d e f g h j k l m n o p q r s t u
	04	a b c d e f g h j k l m n o p q r t u
	05	a b c d e f g h j k l m o p q r s t u
	06	a b c e f g h j k l m n o p q r s t u
20	01	a b c d e f g h j k l m n o p q r s t u

TABLEAU C

ANNEXE 2

Pour chacun des problèmes étudiés, les données relatives à chacune des pièces seront fournies de la façon suivante :

- première colonne : numéros des machines successivement utilisées lors de la fabrication de la pièce.

- seconde colonne : temps d'occupation des machines correspondantes exprimé en heures.

A - PROBLEMES 5 x 5, 6 x 5, 7 x 5, 8 x 5, 9 x 5 et 10 x 5

N° → des pièces.	1		2		3		4		5		6		7		8		9		10	
	M	T	M	T	M	T	M	T	M	T	M	T	M	T	M	T	M	T	M	T
	1	3	1	5	2	10	2	9	3	2	3	9	2	5	3	4	1	8	2	9
	2	1	2	8	1	4	1	8	2	3	2	6	1	7	2	5	4	8	3	7
	3	5	4	7	3	10	5	1	1	3	5	5	3	4	1	4	3	9	1	7
	4	7	3	5	5	2	3	9	4	3	1	5	4	4	4	2	2	5	4	5
	5	5	5	8	4	4	4	3	5	8	4	1	5	4	5	4	5	3	5	10

B - PROBLEMES 5 x 5, 6 x 6, 7 x 7, 8 x 8 et 9 x 9

Numéro → des pièces	1		2		3		4		5		6		7		8		9	
	M	T	M	T	M	T	M	T	M	T	M	T	M	T	M	T	M	T
	1	3	1	5	2	10	2	9	3	2	3	9	2	5	3	4	1	8
	2	8	3	10	1	9	3	10	1	1	2	1	1	4	1	9	2	7
	3	1	5	8	4	4	1	8	2	3	6	6	4	7	2	5	4	8
	4	4	4	7	3	8	5	10	6	7	4	10	3	2	6	8	6	6
	5	5	2	3	9	10	7	1	4	3	9	5	7	4	5	4	3	9
	6	2	7	5	6	2	9	6	5	7	1	5	6	3	7	9	7	5
	7	7	6	5	8	2	8	9	9	3	7	7	9	9	9	2	8	9
	8	6	8	8	7	9	4	10	8	5	5	1	8	4	8	4	5	3
	9	5	9	4	5	4	6	5	7	6	8	3	5	6	4	8	9	8

VU : Le Président de la thèse,

VU et permis d'imprimer :
TOULOUSE, le

LE PRESIDENT
de l'UNIVERSITE des SCIENCES SOCIALES
DE TOULOUSE,

R. PALLARD

BIBLIOGRAPHIE

[1] Machine de simulation du processus intelligent. Communication du Congrès International de Philosophie des Sciences de ZURICH (1953, Docteur SAUVAN).

[2] Dispositif à mémoire adaptative et active à capacité illimitée. Brevet n° 1381.212 (1961. Docteur SAUVAN).

[3] Thèse de Docteur-Ingénieur (1971. PARIS VI. Maurice GENETE).

[4] Theory of scheduling (1967. Addison Wesley. CONWAY - MAXWELL - MILLER).

[5] Un algorithme optimal pour la gestion des processus en temps réel (Février 1974. RAIRO, J. LABETOULLE).

[6] Panorama des méthodes d'ordonnancement optimal (1974. Université de Bordeaux I. Lucas PUN).

[7] Evaluation de machines et d'algorithmes pour traiter les problèmes combinatoires (1975. Ecole des Mines de SAINT-ETIENNE).

[8] Rapport d'étude sur la mémoire active du Docteur SAUVAN. (SESA).

[9] Le simulateur électronique dénommé "Mémoire active" (SNCF, M. LEMAIRE).

[10] Procédé de recherche de trajet optimal et optimateur correspondant. Demande de brevet n° 73.17.373 (1973. CYBCO S. A., LEDIEU).

[11] Machine informatique pour la recherche d'une solution optimale. Demande de brevet (1974. CYBCO S. A., LEDIEU-ESCHENBRENNER).

[12] Combinateur hybride optimisant. Demande de brevet n° 74.12.262 (1974. SNCF. M. GENETE).

[13] Industrial scheduling (1963. Prentice Hall. MUTH-THOMPSON).

[14] Approches logicielles pour la résolution des problèmes combinatoires en temps réel, thèse de 3ème cycle (1976. Toulouse, B. JULLIEN).

[15] Méthode booléennes en recherche opérationnelle (1970. Dunod, HAMMER et RUDEANU).

[16] Organisation et recherche opérationnelle (1969. Eyrolles, MULLER).